VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ADD-ON INSTRUKCE SÍTĚ AS-INTERFACE PRO DOPRAVNÍK
ADD-ON INSTRUCTIONS OF THE AS-INTERFACE FOR THE CONVEYOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Robert Püschel**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Radek Štohl, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Robert Püschel ***ID:*** 203329

Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Add-on instrukce sítě AS-Interface pro dopravník

POKYNY PRO VYPRACOVÁNÍ:

1. Popište průmyslovou sítí AS-Interface.
2. Proveďte rekonstrukci fyzického modelu dopravníku.
3. Realizujte vybrané funkce jako Add-on instrukce pro 2 typy AS-I masterů fy Bihl-Wiedemann.
4. Napište a vypracujte vzorové řešení vhodné laboratorní úlohy pro demonstraci diagnostiky sítě pomocí Add-on instrukcí.
5. Vyhodnoťte své řešení.

DOPORUČENÁ LITERATURA:

Becker, R. a kol.: AS-Interface, řešení pro automatizaci. AS-International Association, 2004, 184 s. ISBN 80-21--2958-5

Dle vlastního literárního průzkumu a doporučení vedoucího práce.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: Ing. Radek Štohl, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

Abstrakt
Cílem bakalářské práce je seznámit čtenáře s tvorbou Add-On instrukcí ve vývojovém prostředí Studio 5000 pro průmyslovou síť AS-Interface, na kterou je napojen model dopravníku. Práce je rozdělena na teoretickou část, zabývající se popisem sítě AS-Interface, fungování Add-On instrukce a modelu dopravníku, a praktickou část, která popisuje konkrétní tvorbu Add-On instrukcí, jejich logiku, demonstraci v úloze a úplný výčet.

Klíčová slova
Add-On instrukce, AS-Interface, PLC, dopravník, Studio 5000

Abstract
The aim of the bachelor‘s thesis is to acquaint readers with a creation of Add-On Instructions in the development environment Studio 5000 for the industrial network AS-Interface, to which the conveyor model is connected. The thesis is divided into a theoretical part dealing with the description of the AS-Interface network, the operation of the Add-On instruction and the conveyor model and a practical part that describes the specific creation of Add-On instructions, their logic, demonstration in a task and a complete list.

Keywords
Add-On instruction, AS-Interface, PLC, Conveyor, Studio 5000

Bibliografická citace

PÜSCHEL, Robert. Add-on instrukce sítě AS-Interface pro dopravník [online]. Brno, 2020 [cit. 2020-05-18]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/126874. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Radek Štohl.

Prohlášení autora o původnosti díla

„Prohlašuji, že svou bakalářskou práci na téma Add-On instrukce sítě AS-Interface pro dopravník jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních, a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI, díl 4 trestního zákoníku č. 40/2009 Sb.“

V Brně dne **8. června 2020** ……

podpis autora

Poděkování

Děkuji Ing. Radku Štohlovi, Ph.D., za odborný a přátelský přístup věnovaný po celou dobu tvorby. Velmi si cením jeho pomoci a všech rad, bez kterých by tato práce nemohla vzniknout.

V Brně dne **8. června 2020** ……

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 ÚVOD

Cílem práce je seznámení čtenáře s komunikační sítí AS-Interface, pomocí které jsou obsluhovány senzory a aktuátory na modelu dopravníku, a tuto komunikaci demonstrovat skrze Add-On instrukce vytvořené v programovém prostředí *Studio 5000*. Rovněž jsou zde popsány samotný dopravník a jeho hardwarová výbava.

AS-Interface je komunikační rozhraní nízké úrovně, protože zajišťuje výměnu informací přímo s jednotlivými slejvy, což jsou buď přímo senzory a akční členy, nebo moduly, ke kterým jsou připojeny. Nahrazuje tedy klasické I/O karty pro PLC a šetří počet použitých kabelů na propojení. Tato síť vznikla hlavně kvůli finanční atraktivnosti, kterou zákazníkovi nabízí. Připojování, konfigurace a správa prvků v této síti jsou totiž maximálně zjednodušeny tak, aby jakékoli poruchy a servisní zásahy mohla zvládnout i nespecializovaná obsluha a provoz nemusel být dlouho pozastaven.

Master AS-Interface umožňuje vysílat do sítě různé požadavky, a vyčítat tak informace z jednotlivých slejvů, nebo naopak informace předávat. Dostáváme se tak k praktické části práce, kde je několik takových příkazů pro síť vytvořeno za použití Add-On instrukcí. Pro práci využívám dvou masterů *BWU1416* a *BWU1488* od firmy Bihl+Wiedemann.

Instrukce jsou realizovány v programu *Studio 5000 Logix Designer*, což je software určený pro PLC produkty od Allen-Bradley, a jsou programovány v jazyce LAD zvoleném pro svou schematičnost a názornost logiky. Na závěr byla vytvořena úloha, která demonstruje využití instrukcí v oblasti diagnostiky připojených slejvů společně s rutinní obsluhou dopravníku. Úloha zároveň testuje funkční koncept instrukcí, ověřuje, zda veškerá výměna dat probíhá v pořádku a zda obsahuje náležité parametry, které uživatel při aplikaci Add-On instrukce využije.

2 TEORETICKÝ ROZBOR

Následující text bude pojednávat o veškerých teoretických přípravách, jež celá problematika vyžaduje nastudovat před započetím práce na samotné realizaci. Čtenář zde bude seznámen s průmyslovou sítí AS-Interface (zkráceně ASi) coby komunikační páteří řešení, modelem dopravníku coby fyzickou reprezentací problému a na závěr Add-On instrukcí coby softwarovou metodou, jak získat požadované informace.

2.1 Průmyslová síť AS-Interface

Síť AS-Interface (ASi), nezkráceně Actuator Sensor Interface, je průmyslovým rozhraním na úrovni fyzické, linkové a aplikační vrstvy dle ISO/OSI, zabývajícím se propojením, jak už z názvu vyplývá, senzorů a akčních členů (aktuátorů). Známou brandovou identitou se stal žlutý profilovaný kabel, který propojuje zařízení se systémem. Stanoveným cílem při vývoji bylo zajištění jednoduchosti, nízké ceny a transparentnosti [1].

Tato kapitola bude mít za cíl popsat, jakým způsobem ASi dosáhlo těchto cílů a kde jsou možné limitace pro uživatele, aby mohl zhodnotit vhodnost nasazení sítě pro svůj konkrétní případ.

Obr. 2-1: Oficiální logo platné od konce roku 2018 [2]

2.1.1 Základní rysy

Za dobu své existence si ASi prošlo čtyřmi iteracemi specifikace. Při uvedení roku 1994 byla verze ASi-2 a o čtyři roky později byla vylepšena na verzi ASi-2.1. Náš model dopravníku pracuje s verzí ASi-3, která šla na trh roku 2004 [3]. Poslední vyvíjenou verzí je ASi-5, která má ambice v takzvaném Průmyslu 4.0, kde jsou více než kdy dříve kladeny vysoké požadavky na digitalizaci [4].

Detailní výpis parametrů všech specifikací znát nepotřebujeme, protože nás zajímá výhradně charakteristika ASi-3, jelikož je užita na modelu dopravníku. Hlavní rysy AS-Interface 3.0:

Tab. 2-1: Hlavní rysy [5]

Topologie sítě	Volná topologie (strom, lineární, hvězdicová, kruhová)
Maximální počet slejvů	62
Sběrnicový kabel	Dvouvodičový kabel pro data a napájení (nestíněný, bez zakončovacích odporů)
Správa sběrnice	Master-slejv s cyklickým pollingem
Maximální doba cyklu	5 ms
Maximální délka sítě	100 m nebo 500 m s opakovači (repeatery)
Standard	IEC 62026-2
Maximální počet I/O	496 vstupů a 496 výstupů
Maximální data na slejva	Vstup 4 bity, výstup 4 bity (cyklicky), několik kilobitů (acyklicky)
Doba odezvy	Typicky 3 ms
Jitter	< ± 150 μs pro synchronizované slejvy
Integrované napájení	24 V DC, max. 8 A
Bezpečnost dat	Zbytková chybovost < 10^{-14} při bitové chybovosti 10^{-4}
Bezpečnostní technologie	Integrovaná
Připojovací technologie	Bez řezání, odizolování a nástrojů
Třída krytí	Až IP69K

Rozhraní vyžaduje ke svému provozu čtyři základní komponenty: slejva, mastera ASi, napájecí zdroj ASi a nadřazený systém (PLC, IPC, Gateway na Ethernet TCP/IP). Z tabulky jsou zřejmá omezení například v délce kabelu nebo množství posílaných dat za cyklus [6].

2.1.2 Topologie

V tabulce 2-1 je zmíněno, že topologie je volná, přesto je dobré si ukázat, jak jednotlivá fyzická uspořádání prvků vypadají a jaké vlastnosti se s nimi pojí. Jejich podobu můžeme vidět na obrázku 2-2 [6].

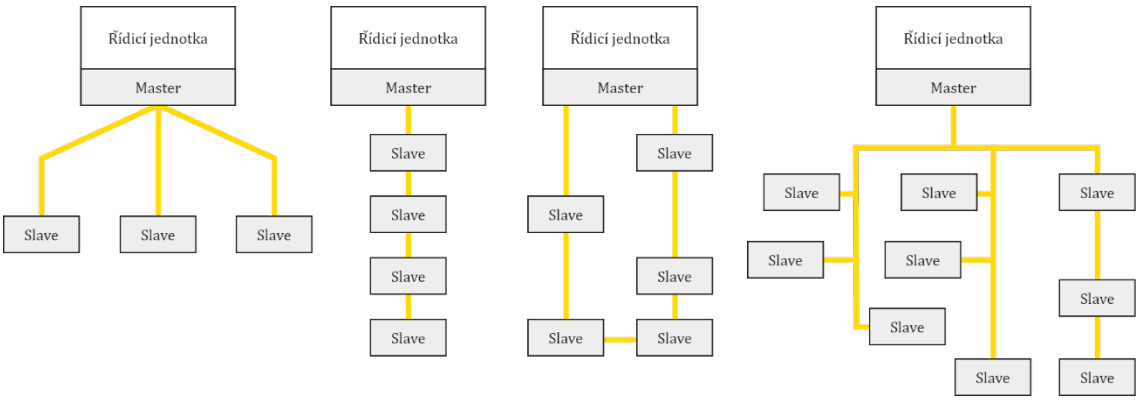
Lineární topologie je propojení, kde jsou všechny prvky sítě na jediné sběrnici. Platí zde pravidla, která jasně určují práva a dobu, kdy který účastník může komunikovat, aniž by narušil probíhající komunikace.

V topologii hvězdy je každý prvek sítě dvoubodově propojen skrze centrální prvek. Díky tomu nemusíme přiřazovat adresy slejvům a netřeba jim definovat pořadí přístupů. Nevýhodou je ovšem neúsporná kabeláž.

Topologie kruhu vzniká, když propojíme sériově veškeré prvky sítě jako u lineární topologie, ale výstup z posledního prvku připojíme na vstup prvku prvního. Díky tomuto uspořádání netřeba dělat adresaci slejvů, jelikož adresu známe díky umístění ve smyčce.

Poslední, topologie stromu, lze dosáhnout odbočkami z jednotlivých ramen sběrnice. Počet odboček a délka nejsou limitovány, a proto tuto strukturu aplikujeme na zařízení, kde chceme lépe pokrýt jeho prostorové možnosti.

Veškeré výše zmíněné topologie lze bezpečně kombinovat, a sestavit tak ideální řešení pro náš projekt. Při projektování bychom ovšem měli uvažovat nejen prostor stroje, ale i způsob, jak chceme organizovat tok zpráv.



Obr. 2-2: Zleva topologie linie, hvězdy, kruhu, stromu

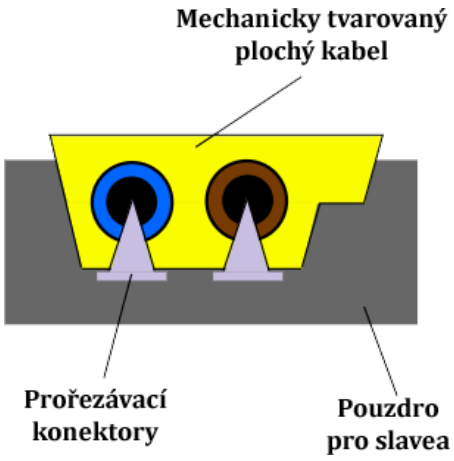
2.1.3 Začlenění do struktury a připojovací technika

ASi propojujeme vždy s vyšší úrovní řízení. Nabízejí se tu dva způsoby [1]:

- **Master ASi jako součást řídicí jednotky.** To znamená, že mastera připojíme k programovatelnému automatu nebo PC. Zároveň se jedná o zapojení užité v našem případě.
- **Master ASi jako součást nadřazené průmyslové sběrnice.** To znamená, že se master společně se slejvy stává gateway (mezisíťovým propojením). Toto rozvržení umožňuje mít více sítí ASi.

Díky tomu, že je síť ASi konfigurována jako soustava PELV *(Protective Extra-Low Voltage)* dle normy *ČSN 33 2000-4-41 ed. 3* [7], odpadá potřeba instalovat ochranný vodič PE a můžeme prvky v rozhraní považovat za bezpečně oddělené od zdrojů napětí a nehrozí přímý ani nepřímý dotyk na živé části pod nebezpečným napětím.

Jak už bylo zmíněno v úvodu kapitoly 2.1, síť ASi je charakteristická svým dvouvodičovým žlutým kabelem. Ten má plochý tvar, aby umožnil prořezávací spoj, aniž by obsluha musela odizolovávat plášť a vodiče. Zároveň je kabel profilovaný tak, aby při připojování modulu nedošlo k obrácení polarity napětí. To umožňuje obsluze velice rychle a jednoduše ke kabelu připojit různé moduly. Celý princip pak znázorňuje obrázek 2-3 [6].



Obr. 2-3: Způsob připojení prvku k ASi [8]

2.1.4 Slejv moduly

V tabulce 2-1 je uvedeno, že k ASi-3 lze připojit až 62 slejvů. Existují dvě možnosti, jak takové připojení provést. Přímé připojení lze realizovat v případě, že snímač nebo akční člen v sobě již obsahují integrovanou elektroniku slejva ASi. Pro uživatele je toto výhodné, jelikož je umožněna efektivní identifikace, parametrizace a diagnostika na úrovni zařízení. Přirozeně ne každý výrobce nabízí námi požadované zařízení s integrovaným slejvem ASi. Druhou možností je tedy připojení přes slejv ASi modul, standardně s konektory M8 nebo M12 popsanými v příloze 1. Některé moduly nabízejí připojení i více než jednoho zařízení, ovšem identifikace, parametrizace a diagnostika jsou stále na úrovni modulu.

Slejv může mít v případě snímače až 8 binárních vstupů, pro aktuátor to je až 8 binárních výstupů. Lze připojit i analogové aktuátory a snímače s rozlišením 16 bitů.

Aby se mohl slejv k síti ASi připojit, a to klidně za provozu, potřebuje mít přidělenu adresu jinou než 0, a to v rozsahu 1 až 31 (1A až 31A) nebo 1B až 31B. Adresa je přidělována podle možností buď adresačním zařízením přes adresační zásuvku, přes IR adaptér, nebo přes konektor. Po spojení s masterem dojde k identifikaci pomocí profilu slejva, jehož stav je často signalizován na diodách [1].

2.1.5 Napájecí zdroj a master

Napájení sítě ASi je konfigurováno jako PELV, což je popsáno v kapitole 2.1.3. Proto nelze využít klasických napájecích zdrojů, ale speciálních, pro ASi určených. Ty tedy musejí splňovat zabezpečení PELV a mají za úkol dodávat napájení 29 až 31 V DC, symetrizovat síť kvůli možnému rušení z okolí a oddělovat data šířící se po vedení.

Master komunikuje se sítí ASi a zároveň s nadřazeným systémem. Master v našem případě je formou karty v PLC, kde nahrazuje klasický modul I/O, kam by se normálně vedly kabely od snímačů a aktuátorů. Umožňuje díky široké škále funkcí úplnou identifikaci, parametrizaci a diagnostiku všech slejvů a samotné sítě ASi. Veškerá data se shromažďují v registrech obrazů vstupních a výstupních dat a jsou cyklicky aktualizována. Dále tu je registr pro diagnostiku, kde se ukládají chyby v periferiích a příznaky (viz tabulku 2-2). V dalších registrech jsou obsažena konfigurační data, kde se porovnávají profily a požadované parametry pro zjištění shody [6].

Tab. 2-2: Příznaky o stavu sítě, odesílané masterem hostiteli [9]

Config_OK	Shoda požadované a skutečné konfigurace
APF	Příliš nízké napětí nebo pokles napětí při poslední výměně
Periphery_OK	Žádný slejv nehlásí chybu periferie
Configuration_active	Pokud je master v konfiguračním režimu, příznak Config_OK je resetován
NomralOperation_active	Master v normálním cyklickém provozu

Díky jednoduchému a automatickému vyhodnocování lze zjistit, že došlo k poruše slejvu resetováním příznaku *Config_OK*. Obsluha může následně slejva opravit a master zahlásí bezchybnou síť nastavením příznaku opět v *Config_OK* [1].

2.1.6 ISO/OSI model

ASi se realizuje do tří vrstev ze sedmi dle ISO/OSI modelu.

První fyzická vrstva definuje mechanické a elektrické spoje pro přenos informací. Pro ASi jsou zde definovány parametry kabelu, zdroj napětí plnící funkce

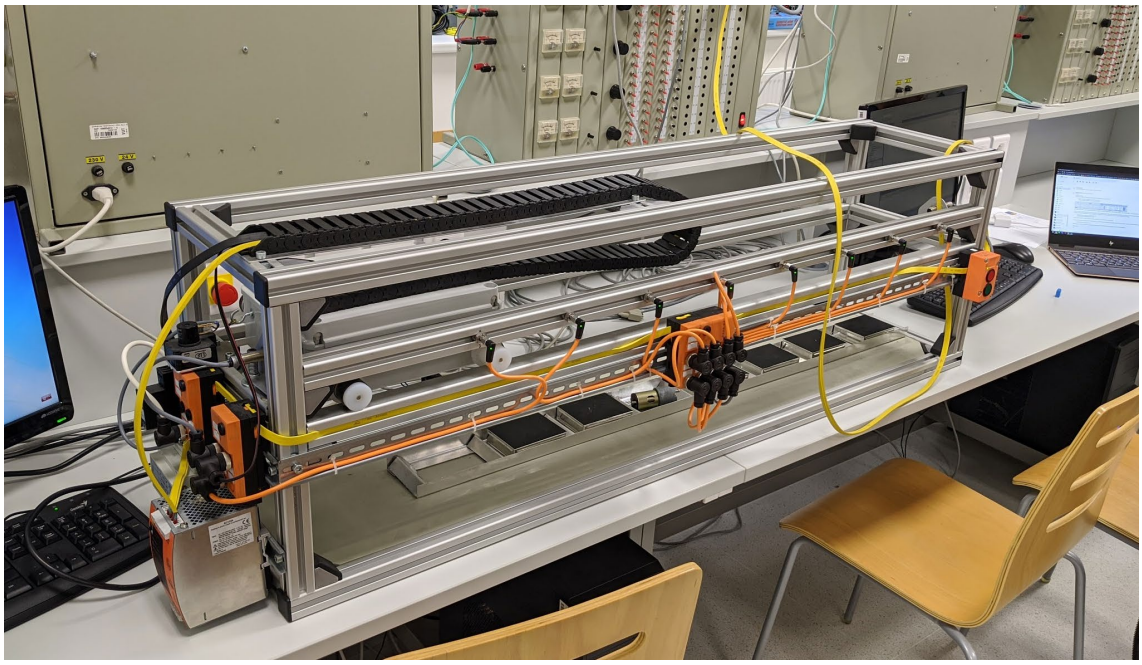
popsané v kapitole 2.1.5 a způsob přenášení dat pomocí střídavě impulzní modulace.

Druhá linková vrstva řeší strukturu dat, rámce, zabezpečení a ošetření chyb. Pojednává, jak musejí vypadat telegramy, které si vyměňuje master se slejvem. Výzva mastera je vždy dlouhá 14 bitů, kde je kromě 5 bitů dat a 4 bezpečnostních a provozních bitů i adresa. Odpověď slejva je vždy 7 bitů, z toho 4 bity dat.

Sedmá aplikační vrstva se týká poskytování služeb uživateli. V ASi to znamená podobu zpráv, jaké cykly probíhají, co znamenají profily slejvů a jak je řešena automatická adresace [6].

2.2 Model dopravníku

Model dopravníku je již vytvořenou částí. Cílem této kapitoly bude pouze popsání jeho konstrukce a osazení zařízeními. Část je věnována i drobným údržbovým zásahům pro zlepšení budoucího chodu.



Obr. 2-4: Reálná podoba modelu dopravníku

2.2.1 Popis konstrukce a blokové schéma

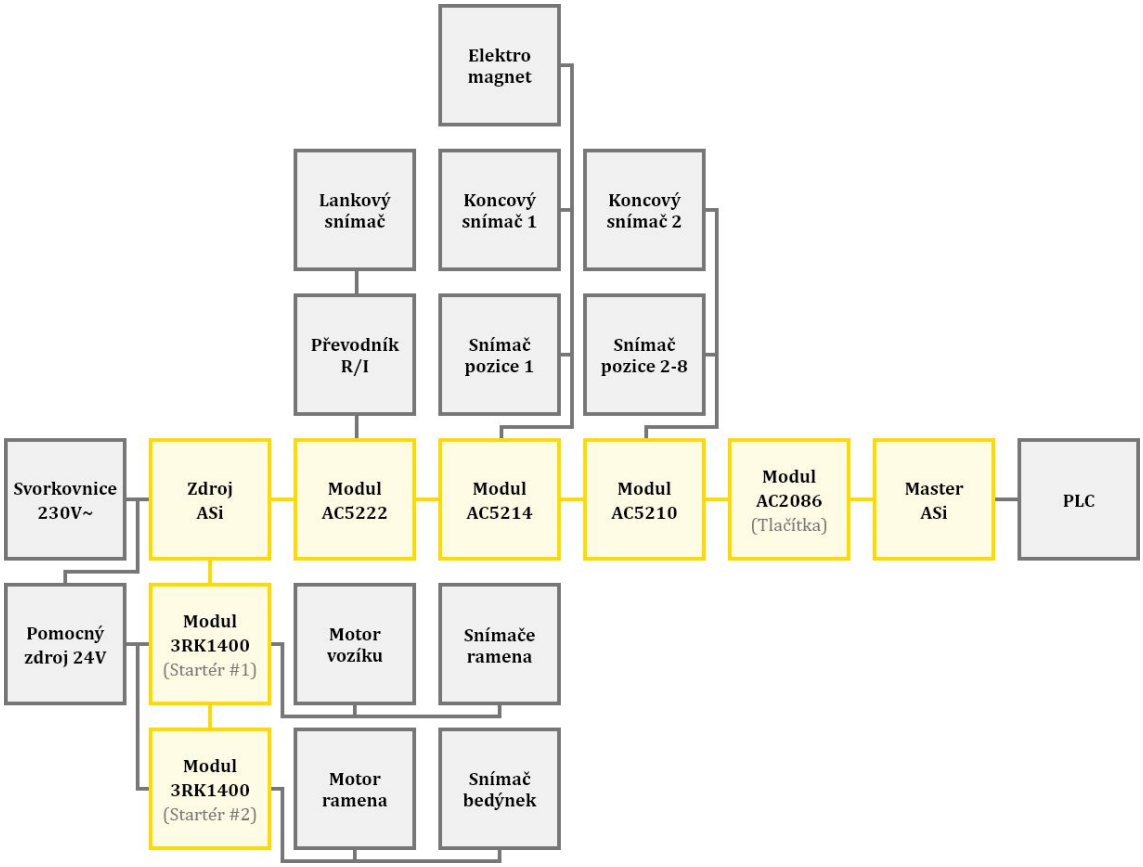
Nosná konstrukce dopravníku je tvořena z profilovaného hliníku rozměru 30 × 30 mm. Celkový kvádrový tvar konstrukce má pak rozměry 1300 × 400 × 300 mm. Na dně modelu se po celé šířce nachází 8 uložení bedýnek, 4 bedýnky jsou přítomny. Přibližně v polovině výšky se nachází kolejnice, po které

se pohybuje vozík s ramenem zakončeným vespod elektromagnetem, pomocí kterého může vozík přemisťovat bedýnky.

Na určení a zajištění polohy vozíku využíváme tří skupin snímačů. První skupinou jsou indukčnostní snímače *871C* na obou koncích kolejnic, čímž se zajistí nevyjetí vozíku z krajních pozic. Druhou skupinou jsou indukčnostní snímače *IEB3004* kolmo nad každým uložením bedýnky, aby bylo možné přesně nad nimi zastavit. Třetí skupinou je odporový lankový snímač *WPS-MK77* v kombinaci s odporovým převodníkem na proud *PX310*, díky kterým můžeme zjistit vzdálenost vozíku od počátku.

Pohyb vozíku zajišťuje motorek s převodovkou *Mega typ 267-0125*, pohyb ramene podobný motorek s převodovkou *Mega typ 267-0781*. Krajní polohu vysunutí ramene hlídají dva indukčnostní snímače *IS5001*. Posledním snímačem je optický snímač *OBT500* ověřující přítomnost bedýnky pod vozíkem.

Všechny snímače jsou připojeny k síti ASi pomocí modulů. Podrobnému popisu prvků na úrovni snímačů a modulů se bude věnovat kapitola 2.2.4. Obecné zapojení nám vyobrazuje blokové schéma na obrázku 2-5 [10].



Obr. 2-5: Blokové schéma modelu dopravníku

2.2.2 Plánované zásahy do konstrukce

Dopravník je třeba zdokonalit primárně v oblasti vozíku. Jako nevhodný se časem ukázal příliš jemný převod hřebenu s ozubeným kolečkem, oba z plastu, a tedy zuby kolečka jsou ojeté, a hřeben tak snadno propadá na dno s těžkým elektromagnetem. Zároveň uchycení elektromagnetu na konci hřebenu je aktuálně řešeno zapuštěním do lepidla, což není dostatečně pevné řešení, jelikož napětím přetěžovaný elektromagnet představuje silnou tepelnou zátěž a dochází k jeho povolování.

Zmíním, že dopravníky existují dva, jeden vždy záložní, a úpravy je třeba provést symetricky na obou. V záložním dopravníku bude dokonce třeba vyměnit motorek pro pohyb vozíku, protože má utrženou jednu přívodní svorku.

Největší výzvu představuje vytvoření nového uchycení motorku ovládajícího rameno vozíku. Hrubší převody znamenají větší rozměry, uchycení tak není použitelné a je třeba objednat zlomek ploché kovové tyče, do které se vyvrtají dva závity pro spojení s vozíkem pomocí šroubů. Na tyč se upevní kování se zavěšeným motorkem a vyfrézovanou drážkou v tyči bude veden hřeben. Nejspíše na samotné krabici vozíku bude třeba vyvrtat nové díry pro šrouby snímačů polohy hřebenu kvůli odskočení z větších rozměrů, abychom zachovali původní otvor, kterým prochází hřeben.

Cílem zásahů nebude nijak měnit původní způsob ovládání nebo podobné principy, s kterými dopravník v minulosti vznikl. Jedná se spíše o nutné zásahy, aby byl dopravník vůbec ovladatelný a dlouhodobě spolehlivý.

2.2.3 ASi hardware, snímače a aktuátory

Na dopravníku se nachází celkem šest ASi slejv modulů, na nichž je připojeno sedmnáct snímačů a aktuátorů. ASi mastera v rámci adres zajímá ovšem pouze počet slejvů. Z tabulky 2-3 lze vyčíst adresaci.

Tab. 2-3: Adresace ASi slejvů

ASi slejv (modul)	Stručný popis I/O	Adresa
3RK1400-1MQ03-0AA4 #1	Startér, 4 × DI, 1 × DO	1
3RK1400-1MQ03-0AA4 #2	Startér, 4 × DI, 1 × DO	2
AC2086	2 × tlačítko s LED	3A
AC5214	2 × DI, 2 × DO	5A
AC5210	8 × DI (2 slejvové)	6A/7A
AC5222	2 × AI	10

Obr. 2-6:
Modul 3RK1400-1MQ03-0AA4 [11]

Obr. 2-7:
Modul AC2086 [12]

Obr. 2-8:
Modul AC5214 [13]

Obr. 2-9:
Modul AC5210 [14]

Obr. 2-10:
Modul AC5222 [15]

Na prvním modulu *3RK1400-1MQ03-0AA4* od společnosti Siemens je připojen na výstup motorek s převodovkou pro vozík a na vstupy dva snímače polohy ramena. Na druhém modulu je zase připojen na výstup motorek s převodovkou pro rameno a na vstup snímač bedýnek.

Modul *AC2086* od společnosti ifm electronic obsahuje dvě tlačítka, tj. dva vstupy. Pod barevnou krytkou tlačítek je ještě LED signalizace, tj. dva výstupy. Modul typicky slouží k uvedení dopravníku do chodu a k jeho zastavení.

Na modul *AC5214* na vstupech jsou připojeny jeden koncový snímač a jeden snímač pozice nad uložením bedýnky. Na výstupu je pak připojen elektromagnet, který je navržen na 12 V, ovšem je přetěžován 24 V, a můžeme jej tak sepnout maximálně na 45 sekund [10].

Modul *AC5210* má na všech osmi vstupech připojeno sedm snímačů pozice a jeden koncový snímač. Snímače jednoho druhu, například snímače pozice, nejsou pospolu na jednom slejvovi z prostorových důvodů modelu dopravníku, proto tyto důvody zapojení na modulech *AC2514*/*AC5210*.

Na modul *AC5222* na jeho analogový vstup je připojen převodník odporu na proud, na který je zase připojen lankový snímač vzdálenosti vozíku. Jedná se o jedinou analogovou informaci z hlediska senzorů/aktuátorů.

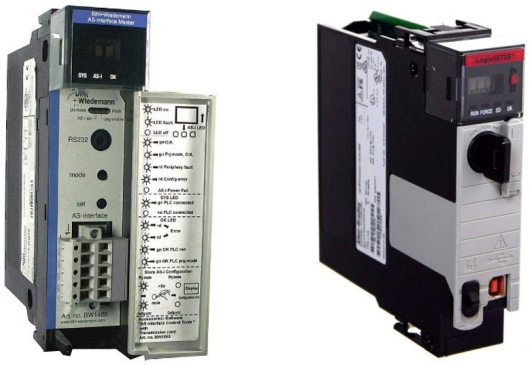
2.2.4 Master hardware a PLC

Cílem zadání je pracovat se dvěma různými ASi mastery. Dopravník tak na sobě nemá pevně zabudovaného mastera, ale dvouvodičový kabel ASi je zakončen konektorem zabraňujícím přepólování, který lze připojit k vybranému masterovi. Jak bylo zmíněno v kapitole 2.1.5, naši masteři jsou kartou v PLC, kde nahrazují klasické I/O moduly.

První ASi-3 master *BWU1416* od společnosti Bihl+Wiedemann je výhradně určen pro PLC od Allen-Bradley *CompactLogix* / *MicroLogix 1500*. My disponujeme modelem Allen-Bradley *CompactLogix 1769-L33ERM*.

Obr. 2-11: Vlevo master BWU1416 [16], vpravo PLC ControlLogix 1769-L33ERM [17]

Druhý ASi-3 master *BWU1488* je výhradně určen pro PLC od Allen-Bradley *ControlLogix*. Oproti předchozímu masterovi k tomuto lze připojit dvě ASi sítě, ne pouze jednu. U nás je tento master v PLC od Allen-Bradley *GuardLogix 1756-L72S*.



Obr. 2-12: Vlevo master BWU1488 [18], vpravo PLC GuardLogix 1756-L72S [19]

2.2.5 Napájení

Aby model dopravníku fungoval, je nutné využít dva napájecí zdroje. Prvním je ASi napájecí zdroj *115/230VAC 4A AC1224* od společnosti ifm electronic. Výstupem je napětí 29,5 až 31,6 V DC a zajišťuje napájení pro moduly a splňuje požadavky popsané v kapitole 2.1.5.

Druhý napájecí zdroj *JS-55-240/DIN* od společnosti BKE je pomocný pro startérové moduly, na které jsou připojeny motorky. Poskytuje na výstupu napětí 24 V DC a proud až 2,3 A.

Obr. 2-13: Vlevo ASi zdroj AC1224 [20], vpravo pomocný zdroj JS-55-240/DIN [21]

2.3 Add-On instrukce

Add-On instrukce je velice účinnou předností softwaru *Studio 5000* a jeho předchozích verzí. Umožňuje programátorovi zabalit do jedné instrukce běžné a často opakovaně používané funkce a algoritmy, což zpřehledňuje a zjednodušuje výsledný program z uživatelského hlediska.

Přestože se Add-On instrukce využívá pro zabalení znovu použitelných algoritmů, není programátorsky vhodné do ní schovávat hlavní rutinu programu. Její uplatnění hledáme v místech, kde potřebujeme využít modularity, kde můžeme vzít určitou logiku algoritmu, kterou lze opakovaně nasazovat do dalších programů.

Instrukci můžeme snadno exportovat a importovat. Mnoho výrobců tvoří ke svému hardwaru právě takové Add-On instrukce, na kterých je obsažena například diagnostika, a uživatel tak má usnadněnou část implementace softwaru.

Dále si z praktického hlediska mohou nést informace o revizích, aby byl jasný přehled o provedených změnách. Jednotlivé revize lze digitálně podepsat a brání se tak proti možné úpravě. To se může zdát zbytečné, ale je důležité myslet na to, že takový software je často nasazován do strojů, na kterých závisejí drahá výroba nebo bezpečnost pracovníků.

Pro zvýšení modularity lze Add-On instrukce zanořovat do sebe tak, že uvnitř jedné se volá další. V našem prostředí toto zanoření může být až do 7. úrovně [22].

3 PRAKTICKÁ ČÁST

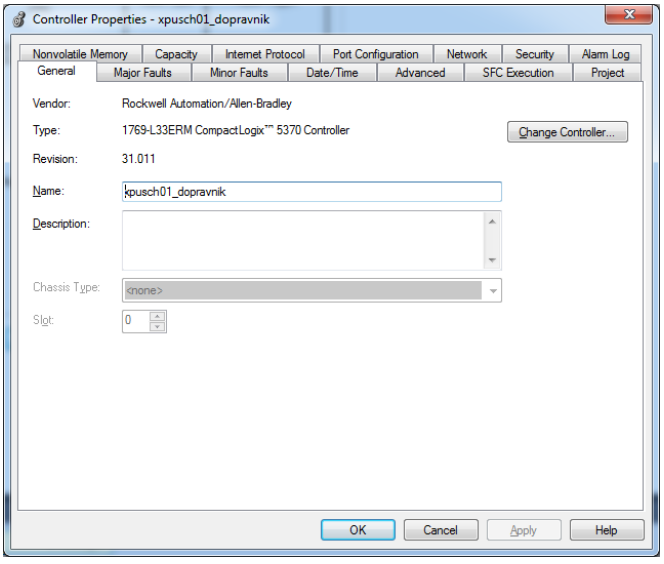
Obsah praktické části vysvětluje postup a návrh tvorby Add-On instrukcí ve vývojovém prostředí *Studio 5000*. Jsou zde popsány nejen logika a výčet realizovaných Add-On instrukcí, ale hlavně postup při konfiguraci vývojového prostředí tak, aby čtenář mohl nasimulovat stejné podmínky a komunikace mu fungovala správně.

3.1 Příprava vývojových prostředí

Jak bylo zmíněno dříve, pro PLC od Allen-Bradley byl použit vývojový nástroj *Studio 5000*. Po spuštění a založení projektu je nutno vybrat správnou hardwarovou konfiguraci. V našem případě se budou vytvářet dvě konfigurace, protože jsou použita dvě odlišná PLC s různými ASi mastery.

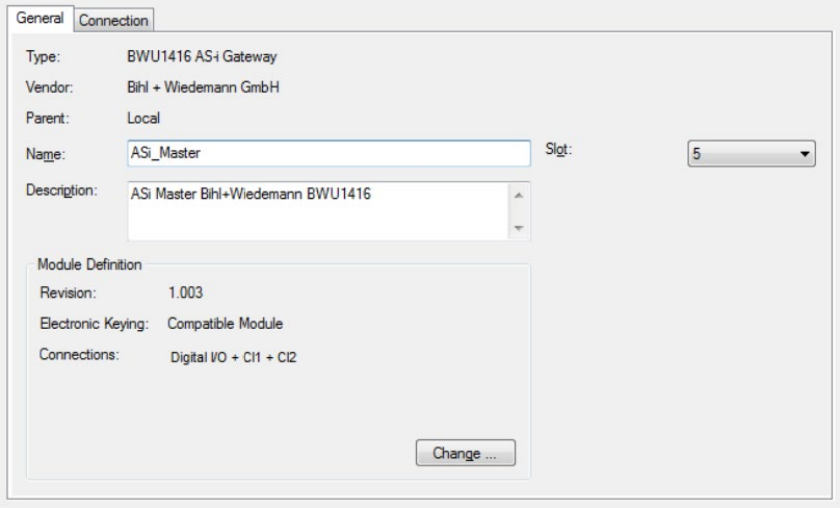
Pomocí programu *RSLinx* zjišťujeme potřebné informace o procesoru, modulech, revizích (firmwaru), použitých slotech v šasi a podobně. Těchto informací budeme využívat dále v postupu.

První popsaná konfigurace PLC bude s *CompactLogix 1769-L33ERM*. Tento zmíněný procesor je třeba vložit v sekci *Controller Organizer* do *I/O Configuration*. Kontroler pojmenujeme a nastavíme správnou instalovanou revizi, v našem případě 31.011. Na obrázku 3-1 lze vidět detail námi vloženého kontroleru.



Obr. 3-1: Konfigurace kontroleru CompactLogix 1769-L33ERM

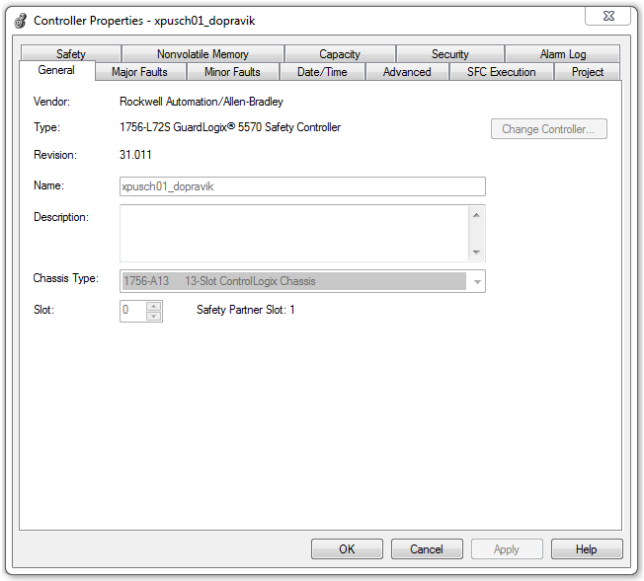
Nyní zbývá obdobně vložit modul s masterem *BWU1416 AS-i Gateway* od výrobce Bihl+Wiedemann. Dáme mu název, vybereme správnou revizi a slot, ve kterém je modul fyzicky instalován. Výslednou konfiguraci lze vidět na obrázku 3-2.



Obr. 3-2: Konfigurace modulu mastera BWU1416

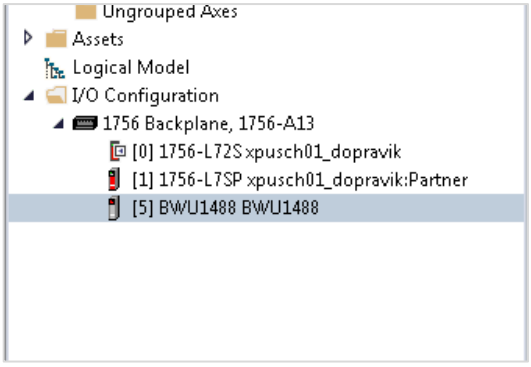
Tímto je příprava prvního prostředí hotová a je třeba konfiguraci nahrát do PLC. V horní části programu *Studio 5000* nalezneme oblast s nápisem „Path“, kde vybereme síťovou cestu k našemu kontroleru. Jakmile tak učiníme, stačí konfiguraci stáhnout pomocí tlačítka *Download* do PLC. Pozor na tlačítko *Upload*, které naopak přenese aktuální konfiguraci v kontroleru do našeho prostředí.

Druhá popsaná konfigurace PLC s *GuardLogix 1756-L72S* bude o něco složitější. Postup je obdobný, ovšem u tohoto kontroleru budeme kromě revize, v našem případě 13, vybírat i *Chasis Type*, šasi, která je 13slotová *1756-A13*.



Obr. 3-3: Konfigurace kontroleru GuardLogix 1756-L72S

Vkládaný modul ASi masteru *BWU1488* nalezneme pod stejnojmenným názvem od výrobce Bihl+Wiedemann. Modul pojmenujeme, vybereme slot 5, kde se fyzicky nachází, a zvolíme revizi, která je 1.001. Jak vypadá importovaný modul mastera ve vývojovém prostředí, ukazuje obrázek 3-4.



Obr. 3-4: Importovaný modul mastera BWU1488

Hotovou konfiguraci projektu můžeme opět stáhnout do PLC. Je dobré zmínit, že nelze držet v rámci jednoho projektu různé konfigurace. To znamená, že naše dvě nastavená prostředí držíme jako dva separátní projekty. Pokud jsme vše provedli správně, při otevření položky *Controller Tags* v sekci *Controller Organizer* nalezneme získávaná vstupní, výstupní a stavová data z ASi masteru.

3.2 Návrh Add-On instrukcí

Naším cílem je vytvořit Add-On instrukce pro síť AS-Interface, která vyčte informace o slejvech dopravníku. Funkce jednotlivých Add-On instrukcí vychází z dostupných příkazů, které lze do sítě poslat a na jejich základě potřebná data číst, nebo naopak data přenášet.

V prostředí programu *Studio 5000* lze přidat novou Add-On instrukci v záložce *Assets – Add-On Instructions* v sekci *Controller Organizer*. Při zakládání je na uživateli, v jakém jazyce se rozhodne logiku psát. V našem případě byl zvolen jazyk *ladder diagram* (LAD). Výsledkem je sice prostorově objemnější logika například oproti *strukturovanému textu* (ST), protože je tvořena grafickými bloky, ale má zase lepší schematickou názornost řešeného problému.

3.2.1 Popis chování

Z vnějšího pohledu lze instrukci popsat pomocí vstupních a výstupních parametrů, kterým se věnuje kapitola 3.2.2, ovšem níže na obrázku 3-5 si už ukážeme její obecnou blokovou podobu z pohledu uživatele.

Obr. 3-5: Add-On instrukce v blokové podobě

Z vnitřního pohledu Add-On instrukci již popíšeme logikou, která má na starost odeslání požadavků (příkazu, výběru sítě, způsobu uspořádání bitů, přepnutí toggle bitu, případně dalších parametrů) a vyčtení odpovědí (zrcadleného příkazu, výstupu toggle bitu, kódu odpovědi, případně dalších odpovědí).

Ještě před ukázkou vývojového diagramu popisujícího chování instrukce je dobré si ukázat podobu konstrukce příkazového rozhraní, abychom pochopili důvody logiky. To se rozděluje na *Request* („Požadavek“, obrázek 3-6 a popis pod ním) a *Response* („Odpověď“, obrázek 3-7 a popis pod ním).

command request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	command							
2	T	O	circuit					
3	request parameter byte 1							
…	…							
36	request parameter byte 34							

Obr. 3-6: Konstrukce požadavku „Request“ ASi [23]

- **Byte:** Počet bytů požadavku, kde 36 je maximální délka. Byty 1 a 2 jsou přítomny vždy, byty 3 až 36 záleží na druhu požadavku.
- **Command:** Kód instrukce.
- **T:** Toggle bit, který se přepíná dle provedení instrukce. Slouží jako zpětná vazba při cyklickém přenosu dat.
- **O:** Order bit, který říká, jak budou uspořádány bity v rozhraní.
  - **= 0:** Vybere standardní schéma (kde 2^0 je nejnižší bit)
  - **= 1:** Vybere Siemens schéma (kde 2^7 je nejnižší bit)
- **Circuit:** Výběr ASi sítě. Některá rozhraní masterů umožňují připojit až dvě sítě ASi (například u *BWU1488* lze mít dvě). **Circuit = 0** je výběr první sítě.
- **Request parameter byte n**: Je ‚n‘ parametrů příkazu. Pokud příkaz nevyžaduje zadat parametry na daném bytu, tak je netřeba nulovat, budou jednoduše ignorovány.

command response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	command (mirrored)							
2	T	result						
3	response byte 1							
…	…							
36	response byte 34							

Obr. 3-7: Konstrukce odpovědi „Response“ ASi [23]

- **Byte:** Počet bytů odpovědi, kde 36 je maximální délka. Byty 1 a 2 jsou přítomny vždy, byty 3 až 36 záleží na druhu odpovědi.
- **Command (mirrored):** Zrcadlený kód instrukce.
- **Result:** Kód odpovědi. Významu kódů se věnuje tabulka 3-1.
- **Response byte n**: Je ‚n‘ bytů odpovědi.

Tab. 3-1: Kódy odpovědí Add-On instrukce [23]

Název	Hodnota HEX	Popis
OK	00	Provedeno bez chyby
HI_NG	11	Obecná chyba
HI_OPCODE	12	Neplatná hodnota v příkazu
HI_LENGTH	13	Délka příkazového rozhraní je příliš krátká
HI_ACCESS	14	Bez přístupových práv
EC_NG	21	Obecná chyba
EC_SND	22	Slejv (zdrojové adresy) nenalezen
EC_SD0	23	Slejv 0 nalezen
EC_SD2	24	Slejv (cílové adresy) nenalezen
EC_DE	25	Chyba mazání (delete)
EC_SE	26	Chyba nastavení (set)
EC_AT	27	Dočasná adresa
EC_ET	28	Dočasně prodlouženo ID1
EC_RE	29	Chyba čtení (prodlouženého ID1)

Programově pracujeme právě s těmi hodnotami, které jsme popsali v kapitole 3.1 o nastavení ASi modulů masterů a získání potřebných vstupů a výstupů v *Controller Tags*. Na výstup (O: *Output*) zapisujeme požadavky a ze vstupu (I: *Input*) čteme odpovědi. Přesné umístnění úseků, kam zapisovat a odkud číst, je uvedeno v příloze 2.

Obr. 3-8: Vývojový diagram Add-On instrukce

3.2.2 Parametry a lokální tagy

Při vytváření Add-On instrukce musíme nadefinovat její parametry a lokální tagy. Parametry reflektují vnější, tedy vstupně-výstupní popis. Říkají nám, co můžeme nebo musíme instrukci předat a kam to má zapsat. Lokální tagy naopak reflektují vnitřní popis, tedy jaké proměnné instrukce potřebuje na provoz své logiky.

Výčtu parametrů a lokálních tagů se věnují tabulky 3-2 a 3-3 a zároveň popisují i vzájemné odchylky při implementaci pro dva různé mastery. Tabulka parametrů ovšem nereflektuje potřeby všech Add-On instrukcí, ale pouze jejich průnik. Některé příkazy totiž vyžadují ke své činnosti i další vstupní parametry, jako jsou třeba adresy slejvů.

Tab. 3-2: Parametry Add-On instrukce

Parametry					
Název	**Datový typ**	**Použití**	**Vyžadován**	**BWU1416**	**BWU1488**
EnableIn	BOOL	Input		×	×
EnableOut	BOOL	Output		×	×
ASi_Input	Vstupy ASi modulu	InOut	×	×	×
ASi_Output	Výstupy ASi modulu	InOut	×	×	×
Result	SINT[36]	InOut	×	×	×
Result_Length	SINT	Output		×	×
Done	BOOL	Output		×	×
Run	BOOL	InOut	×	×	×
Circuit	SINT	Input			×

- **EnableIn:** Pevný parametr funkce, povoluje její spuštění.
- **EnableOut:** Pevný parametr funkce, je zpětnou vazbou k vykonání instrukce v zavislosti na EnableIn.
- **ASi_Input:** Čtené vstupy ze sběrnice z modulu mastera ASi. Datový typ je závislý na druhu modulu a jeho parametrů.
- **ASi_Output:** Zapisované výstupy na sběrnici na modul mastera ASi. Datový typ je závislý na druhu modulu a jeho parametrů.
- **Result:** Proměnná, kam se uloží odpověď vyčtená ze vstupů modulu mastera ASi. Délka datového typu je dána maximální možnou délkou odpovědi, viz obr. 3-7.
- **Result_Length:** Informuje uživatele o skutečné délce odpovědi v závislosti na druhu instrukce (ne každá má rozsah 36 SINTů).
- **Done:** Proměnná signalizující provedení instrukce.
- **Run:** Proměnná pro spuštění chodu instrukce.
- **Circuit:** Proměnná pro výběr sítě ASi. Tato proměnná má smysl pouze pro BWU1488, kde můžeme vybírat mezi dvěma sítěmi.

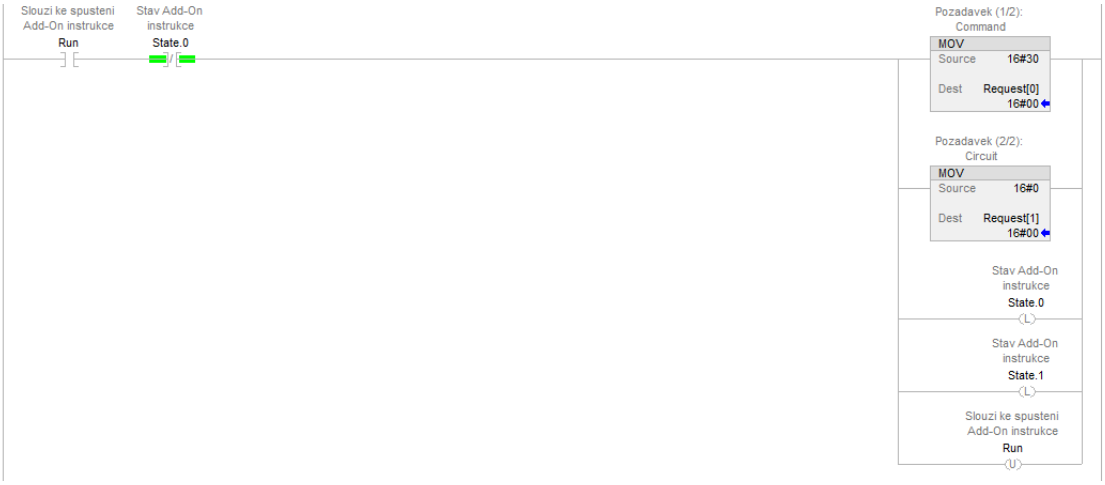
Tab. 3-3: Lokální tagy Add-On instrukce

Lokální tagy		
Název	**Datový typ**	**Popis**
Data	INT[18]	Prostor pro předání a získání hodnot
Byte	SINT	Proměnná pro uspořádání dat z SINT na INT
Request	SINT[36]	Prostor s požadavkem pro ASi
Response	SINT[36]	Prostor s odpovědí od ASi
State	INT	Aktuální stav instrukce

3.2.3 Návrh a konstrukce logiky

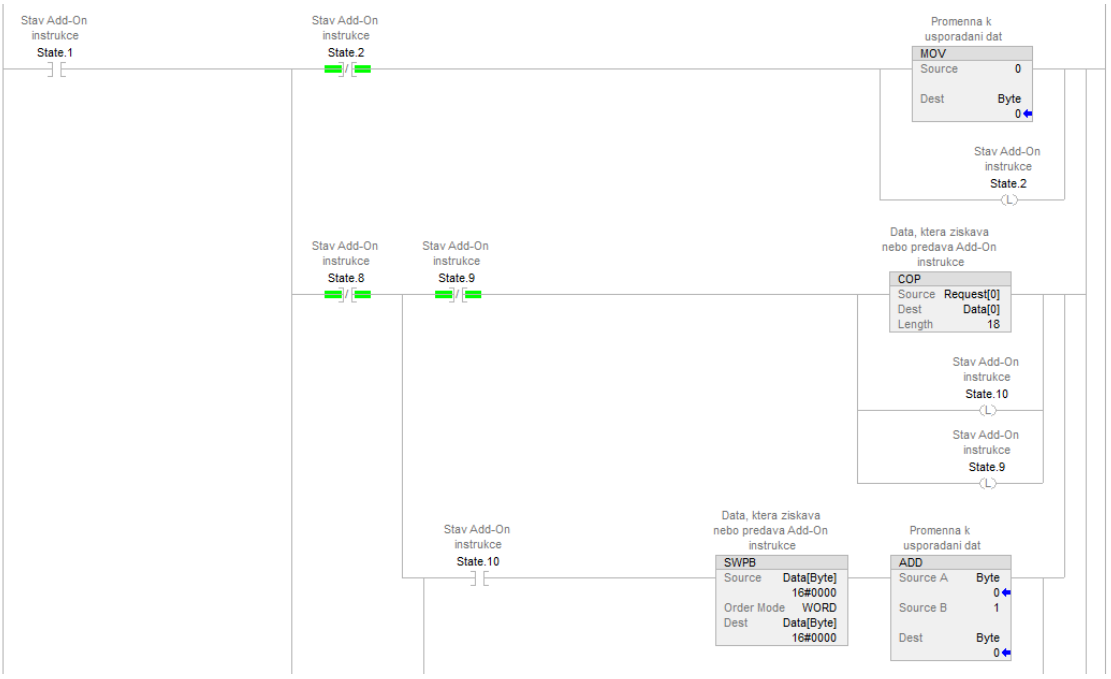
V této kapitole jsou popsány jednotlivé úseky vnitřní logiky instrukce, kde její průběh byl již naznačen na obrázku 3-7. Logika zde bude popsána na příkladu instrukce *GET_LISTS* pro ASi master *BWU1416*.

Spuštění je podmíněno parametrem spuštění *(Run)* od bitu. Následně se do požadavku *(Request)* nahraje příkaz *(Command)* s kódem *16#30* a obvod *(Circuit) 16#00*. Přestože by obvod nemusel být přítomen, z hlediska univerzální konstrukce jej uvádím, zároveň dodržím pomyslné požadavky, co si daná instrukce vyžaduje, přestože vím, že inicializace tagu *Request* nastavila hodnoty na nulu.



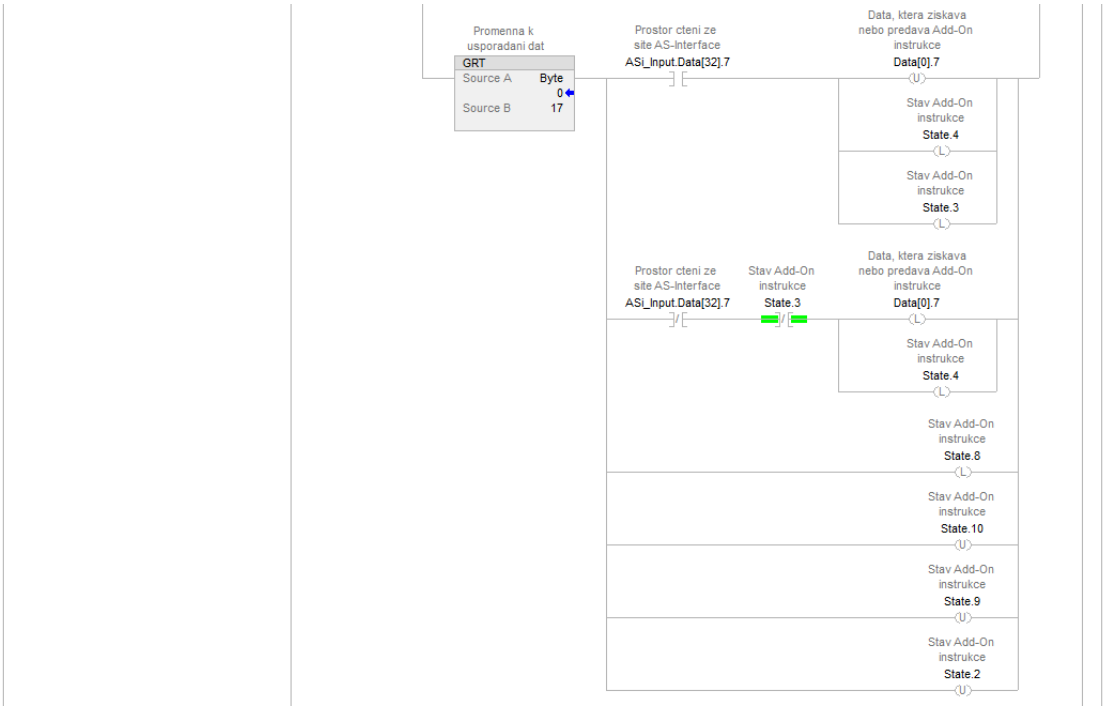
Obr. 3-9: Příklad logiky Add-On instrukce, 1/5

Následuje vynulování proměnné pro uspořádání dat *(Byte)* a nahrání požadavku do dat *(Data)*. Pak je potřeba správně uspořádat jednotlivé bity v datech pomocí bloku *SWAP*, aby odpovídaly 16bitové struktuře *(INT/WORD)*, jak je v dokumentacích k ASi prezentováno. Zde by se dalo opět pojednávat o zbytečnosti kroku, protože I/O *BWU1416* jsou 8bitové *(SINT)* a netřeba přece držet jinou strukturu v datech. Tímto jsem si ale zajistil univerzálnost přípravy dat pro další mastery, jako je například druhý *BWU1488*, který má I/O 16bitové.



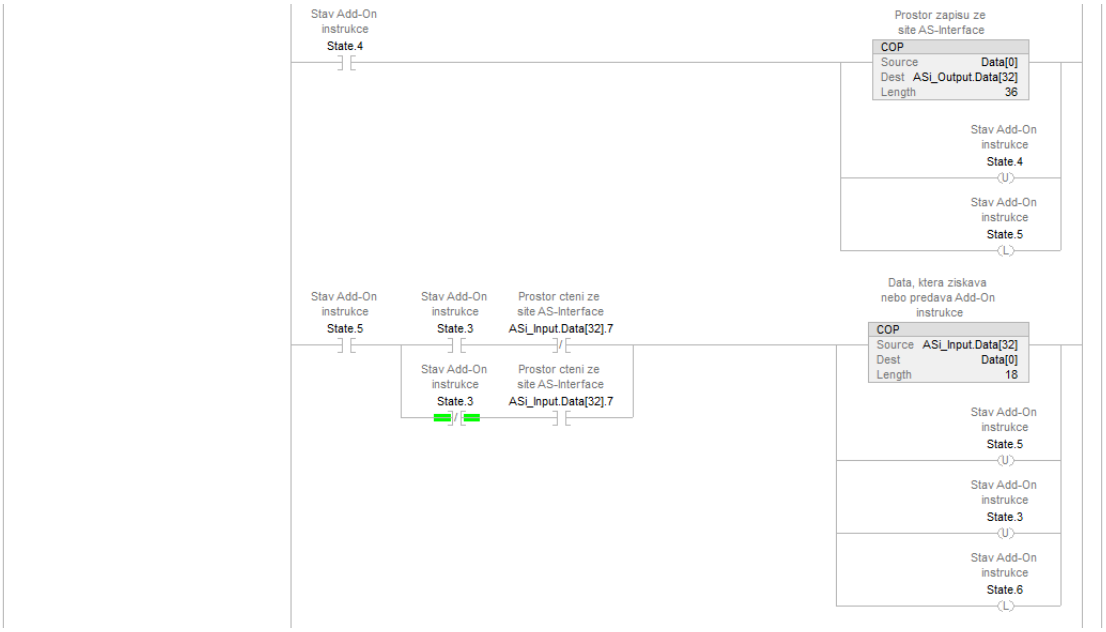
Obr. 3-10: Příklad logiky Add-On instrukce, 2/5

Po uspořádání se na základě toggle bitu ze vstupu ASi nyní rozhodne, zda toggle bit v datech zůstane 0, nebo bude přepnut do 1. Zároveň si do paměti stavu poznamenám, které rozhodnutí proběhlo pro pozdější vyhodnocení.



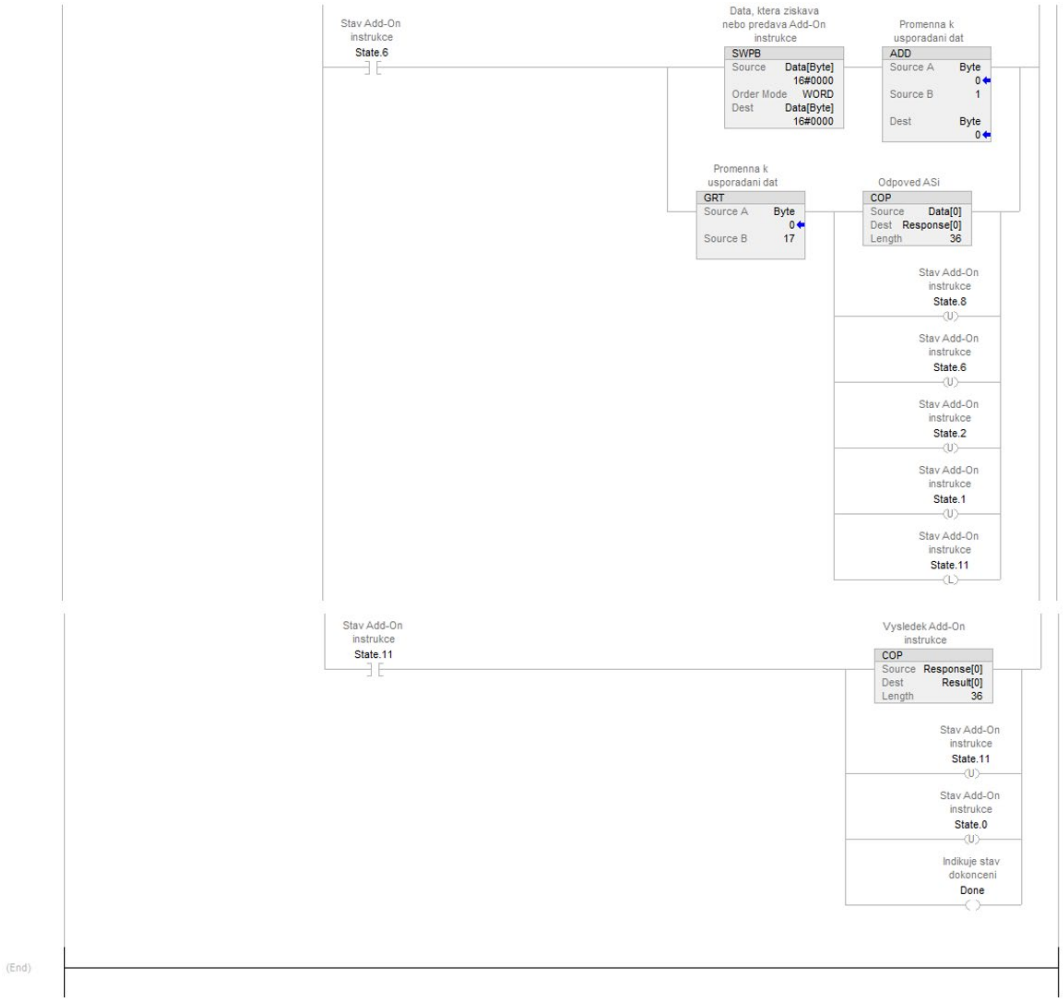
Obr. 3-11: Příklad logiky Add-On instrukce, 3/5

Data jsou následně poslána na výstup ASi. Na vstupu ASi tím pádem došlo ke změně toggle bitu. To vyhodnotím pomocí logiky XOR, zda opravdu došlo ke změně z 0 na 1, nebo z 1 na 0. Jakýkoli jiný stav je chybný a program nebude pokračovat. Při úspěchu jsou vstupy ASi přečteny a zapsány do dat.



Obr. 3-12: Příklad logiky Add-On instrukce, 4/5

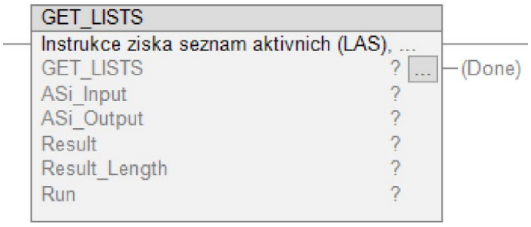
Pomocí bloku *SWAP* opět přeuspořádám bity tak, abych připravil výsledná data do odpovědi *(Response)* v 8bitové struktuře. Odpověď následně pošlu do pro uživatele viditelného výsledku *(Result)* a sepnu příznak dokončení *(Done)*. Instrukce je ve svém výchozím stavu a připravena k dalšímu spuštění.



Obr. 3-13: Příklad logiky Add-On instrukce, 5/5

3.2.4 Implementace a další použití

Po dokončení Add-On instrukce zbývá popsat její implementaci do programu. Při pohledu do horní části *Studia 5000*, kde je oblast se záložkami instrukcí, se přepnu do té s názvem Add-On. Zde nalezneme všechny vytvořené, případně importované Add-On instrukce v projektu a do kódu ji vložím naprosto stejným způsobem jako jakoukoli jinou instrukci. Alternativně ji mohu vložit přetažením přímo ze složky *Add-On Instructions* v sekci *Controller Organizer* do kódu.



Obr. 3-14: Add-On instrukce po vložení do programu

Objeví se blok, jak je vidět na obrázku 3-14, kterému nyní musím přiřadit správné proměnné vyžadovaným parametrům, aby mohl pracovat. Datový typ proměnných musí být shodný s parametry popsanými v tabulce 3-2. Proměnné si buď připravím předem, nebo je vytvořím přímo přes místní nabídku daného parametru v bloku. Hned jako první je třeba přiřadit parametr datového typu Add-On instrukce, který lze vytvořit až s přítomností dané Add-On instrukce v programu, jinak tento datový typ nenalezneme.

Instrukce je navržena tak, že se do programu vkládá paralelně, a to bez obavy, že by byla spuštěna. Je jasně řízena uživatelem pomocí bitu *Run* a signalizuje své dokončení bitem *Done*, jak je zřejmé z její logiky. Podobným způsobem se do programu přidávají třeba časovače a podobně.

Pokud chci mou instrukci exportovat do souboru, lze tak učinit pravým kliknutím na Add-On v sekci *Controller Organizer* a zvolit export. Soubor má příponu souboru *.L5X* a mohu jej libovolně importovat do jiných projektů. Při importu pak stačí znovu správně napárovat parametry a docílím jejich funkčnosti.

3.3 Vytvořené Add-On instrukce

Síť AS-Interface umožňuje uskutečnění desítek různých příkazů, ale jejich základ je vždy stejný. Tato práce se zaměřila na tvorbu diagnostických, analogových a konfiguračních instrukcí, kde vybrané byly využity pro pozdější návrh úlohy, kde je demonstrována funkce Add-On instrukcí v uživatelském programu.

Celkem tak bylo vytvořeno 36 diagnostických Add-On instrukcí pro dva různé mastery, tedy 72 Add-On instrukcí celkem. Jejich výčtu se věnuje příloha 4. V úloze je z toho využito 5 instrukcí, ostatní nebylo třeba pro zadanou situaci používat, a některé instrukce by byly dokonce redundantní. Příkladem může být instrukce GET_LISTS, která získá seznam aktivních, detekovaných a projektovaných slejvů, a tedy použití instrukce GET_LAS, která získá jen seznam aktivních slejvů, je zbytečné, ovšem v některých aplikacích to tak být nemusí, kde nám jde pouze o tuto informaci, a můžeme tak ušetřit data a výpočetní čas.

3.4 Návrh úlohy s diagnostikou dopravníku

Cílem demonstrační úlohy je využít diagnostických Add-On instrukcí ke sledování stavu dopravníku a upozornit, pokud dojde k poruše.

Jeden z modulů dopravníku má do svého M5 konektoru vložen snadno odnímatelný rezistor coby naprojektovanou zátěž, při jeho nepřítomnosti je slejvem vygenerována chyba. Lze vyvolat „závadu“ a pozorovat, jak se zachová program.

Pro úplnost úloha neobsahuje pouze diagnostiku, což je jen jedna ze subrutin, ale i inicializaci a přesouvání bedýnek v nekonečné smyčce řízené uživatelem pomocí tlačítek.

3.4.1 Užité proměnné

Proměnné byly tvořeny v *Controller Tags* v sekci *Controller Organizer*. Alternativně šlo proměnné vytvořit lokálně pouze pro hlavní *(main)* rutinu.

Tab. 3-4: Controller tagy úlohy 1/2

Název	Původní tag	Datový typ	Popis
Local:5:I	–	SINT[XXX] dle BWU	Vstupy mastera ASi
Local:5:O	–	SINT[XXX] dle BWU	Výstupy mastera ASi
Stav	–	SINT	Stav jádra
Stav_D	–	SINT	Stav diagnostiky
Stav_I	–	SINT	Stav inicializace
Stav_A	–	SINT	Stav přesunu z prvního uložení na poslední
Stav_B	–	SINT	Stav přesunu z posledního uložení na první
Konec	–	SINT	Pomocný stav ukočení chodu
Porucha	–	BOOL	Příznak libovolné poruchy
SW_Start	–	BOOL	Pomocný softwarový start
SW_Stop	–	BOOL	Pomocný softwarový stop
Result	–	SINT[36]	Odpověď aktivní Add-On instrukce
Result_DELTA	–	SINT[10]	Odpověď Add-On GET_DELTA
Result_FLAGS	–	SINT[5]	Odpověď Add-On GET_FLAGS
Result_LISTS	–	SINT[29]	Odpověď Add-On GET_LISTS
Result_LOS	–	SINT[10]	Odpověď Add-On GET_LOS
Result_LPF	–	SINT[10]	Odpověď Add-On GET_LPF

Tab. 3-5: Controller tagy úlohy 2/2

Název	Původní tag	Datový typ	Popis
Run_GET_DELTA	–	BOOL	Spuštění Add-On GET_DELTA
Run_GET_FLAGS	–	BOOL	Spuštění Add-On GET_FLAGS
Run_GET_LISTS	–	BOOL	Spuštění Add-On GET_LISTS
Run_GET_LOS	–	BOOL	Spuštění Add-On GET_LOS
Run_GET_LPF	–	BOOL	Spuštění Add-On GET_LPF
Autoflags	Local:5:I.Data[1]	SINT	Automatická detekce příznaků
X_START	Local:5:I.Data[0].2	BOOL	Zelené tlačítko (start)
X_STOP	Local:5:I.Data[0].3	BOOL	Červené tlačítko (stop)
X_C1	Local:5:I.Data[3].3	BOOL	Pozice prvního uložení bedýnky
X_C8	Local:5:I.Data[2].1	BOOL	Pozice posledního uložení bedýnky
X_C10	Local:5:I.Data[0].4	BOOL	Přítomnost bedýnky
X_C11	Local:5:I.Data[1].0	BOOL	Rameno dole
X_C12	Local:5:I.Data[1].2	BOOL	Rameno nahoře
X_ELMAG	Local:5:O.Data[3].0	BOOL	Elektromagnet
X_LED_KO	Local:5:O.Data[0].1	BOOL	LED červeného tlačítka (porucha)
X_LED_OK	Local:5:O.Data[0].0	BOOL	LED zeleného tlačítka (přesun)
X_RAM_DOLU	Local:5:O.Data[1].0	BOOL	Motor ramene dolů
X_RAM_NAHORU	Local:5:O.Data[1].1	BOOL	Motor ramene nahoru
X_VOZ_VPRED	Local:5:O.Data[0].4	BOOL	Motor vozíku vpřed
X_VOZ_VZAD	Local:5:O.Data[0].5	BOOL	Motor vozíku vzad

Uvedení proměnných před popisem logiky je poměrně klíčové pro snadnou orientaci v programu. Obecně lze rozdělit proměnné na funkční (I/O mastera), stavové (rutin) a řídicí Add-On instrukcí.

Tab. 3-6: Parametry a lokální tagy úlohy

Parametry a lokální tagy		
Název	**Datový typ**	**Popis**
GET_DELTA	GET_DELTA	Získání slejvů s konfigurační chybou
GET_FLAGS	GET_FLAGS	Získání příznaků
GET_LISTS	GET_LISTS	Získání aktivních, detekovaných a projektovaných slejvů s příznaky
GET_LOS	GET_LOS	Získaní offline slejvů
GET_LPF	GET_LPF	Získání chyb periferií

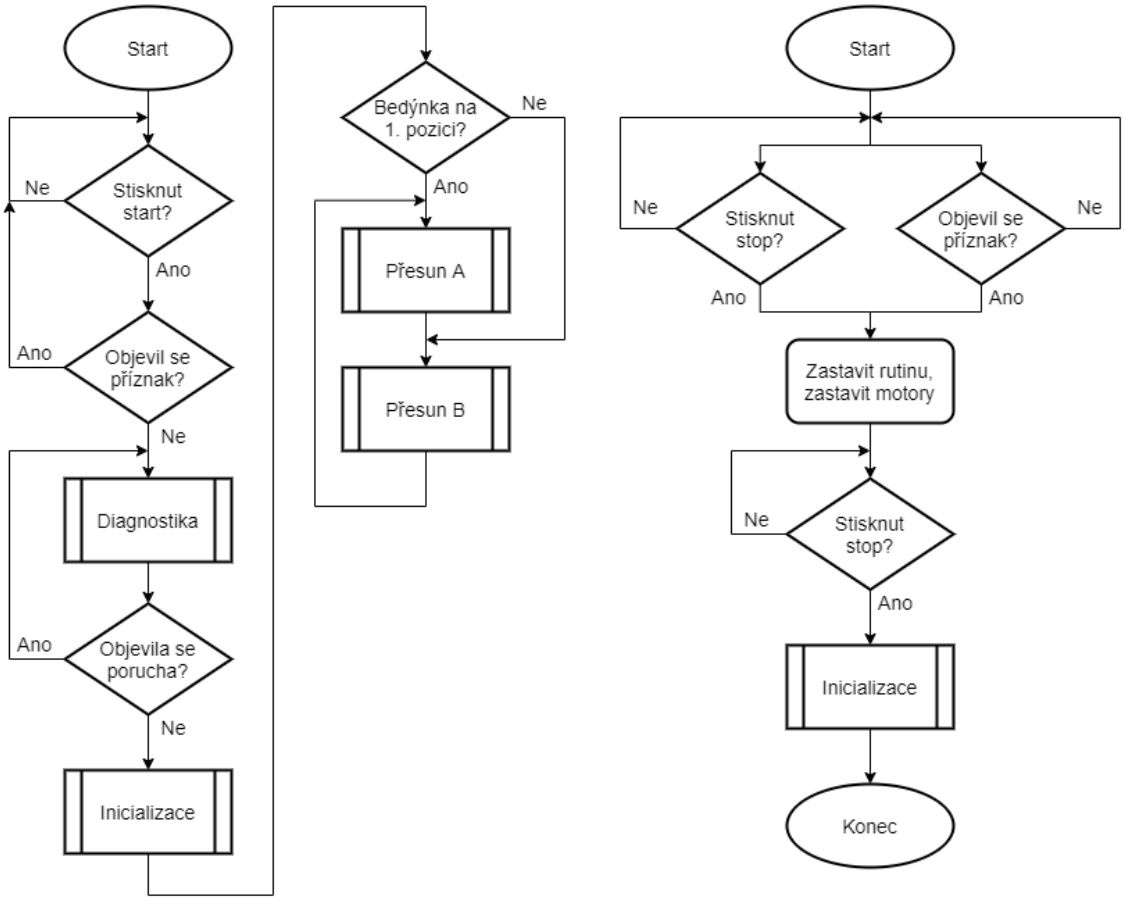
Parametry a lokální tagy byly využity pro vytvoření proměnných datového typu Add-On instrukcí, což je potřeba, abych mohl tyto instrukce nalézt a použít při programování.

3.4.2 Návrh logiky

Program úlohy je rozdělen do hlavní rutiny a čtyř subrutin. Ta hlavní je nazývána jádrem a vyhodnocuje zapnutí a vypnutí chodu a určuje, kdy která subrutina bude spuštěna. Subrutiny jsou diagnostická, inicializační a dvě vzájemné zacyklené přesun beden „tam“ a přesun beden „zpět“.

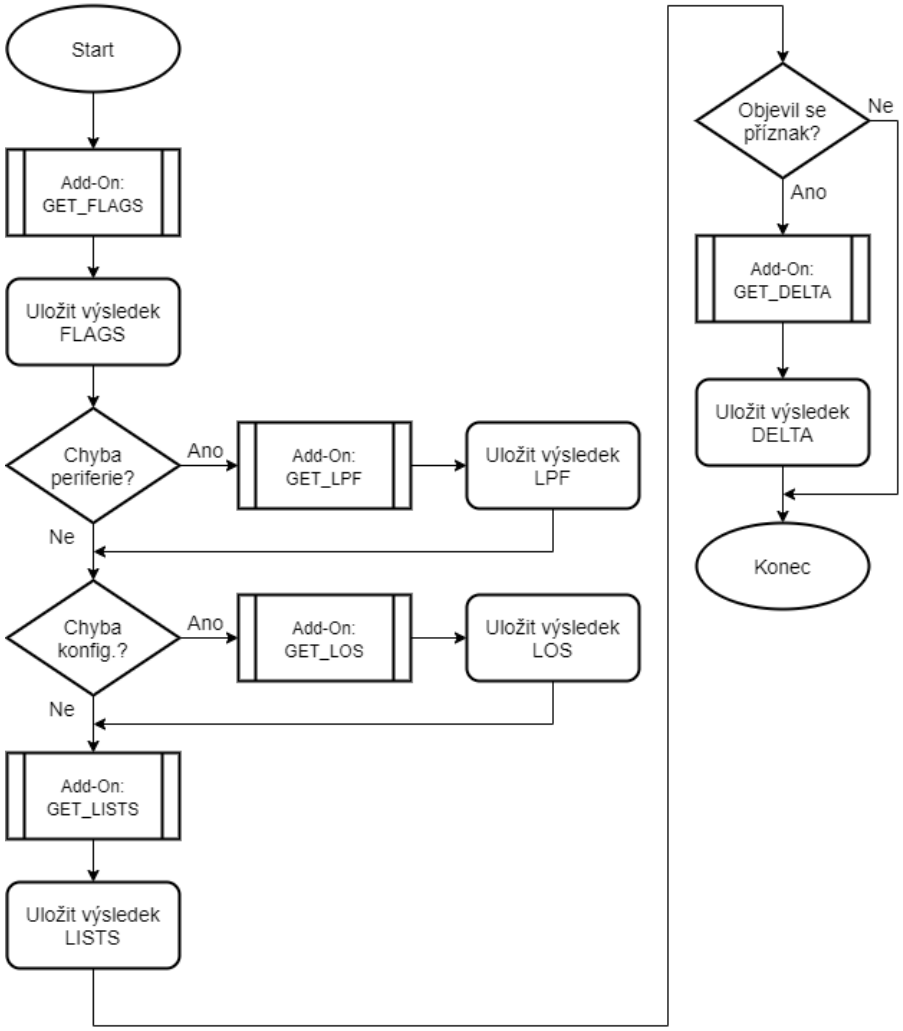
Každá rutina je popsána pomocí vývojového diagramu a později ladder diagramem v další kapitole ‚Konstrukce logiky‘, jak je ve *Studio 5000*, s popisem.

Hlavní rutina ‚Jádro‘ se skládá v podstatě ze dvou paralelně běžících procesů. První proces vyhodnocuje bezchybné spuštění a začne obsluhovat jednotlivé subrutiny. Druhý proces vyhodnocuje chybný příznak nebo stop a na základě toho provede pozastavení chodu. Proces následně čeká na druhý stisk tlačítka stop, aby provedl inicializaci. To je z důvodu bezpečnosti, aby tlačítko stop neuvolnilo sepnutý elektromagnet a zároveň byla zajištěna kontrolovatelná poloha dopravníku.



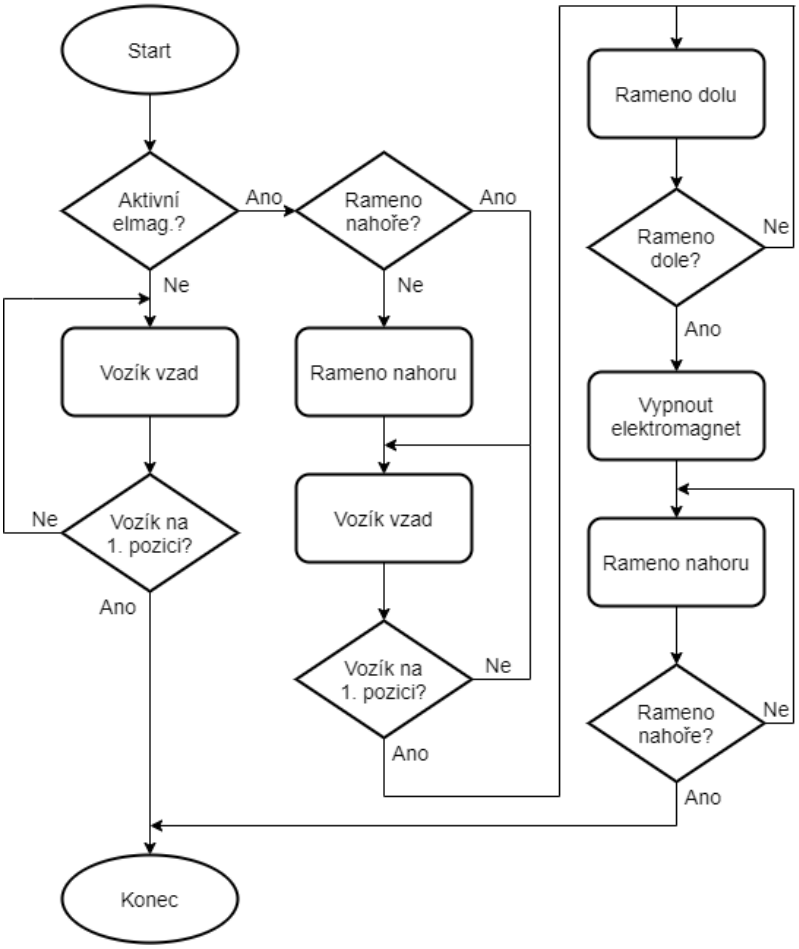
Obr. 3-15: Vývojový diagram hlavní rutiny ‚Jádro‘

Subrutina ‚Diagnostika' je postavena kompletně na využívání informací z Add-On instrukcí. Nejprve se dotáže na obecné příznaky chyb a podle nich, pokud existují, volá další instrukce zjišťující podrobnosti. Instrukce GET_FLAGS a GET_LISTS se volají vždy, protože nemají přímou vazbu na vznik chyby. Popis instrukcí je v příloze 3. Tato subrutina je zároveň klíčovou pro celou bakalářskou práci, protože demonstruje využití vytvořených Add-On instrukcí pro diagnostiku, což je jedním z cílů práce.



Obr. 3-16: Vývojový diagram subrutiny ‚Diagnostika'

Subrutina ‚Inicializace' má za cíl ověřit, že rameno nenese žádnou bedýnku, což by bylo způsobeno aktivním elektromagnetem, a dostat vozík na pozici prvního uložení bedýnky. Pokud je elektromagnet aktivní, inicializace vždy zajistí vrácení bedýnky na volnou pozici, odkud byla bedýnka dříve uchopena.

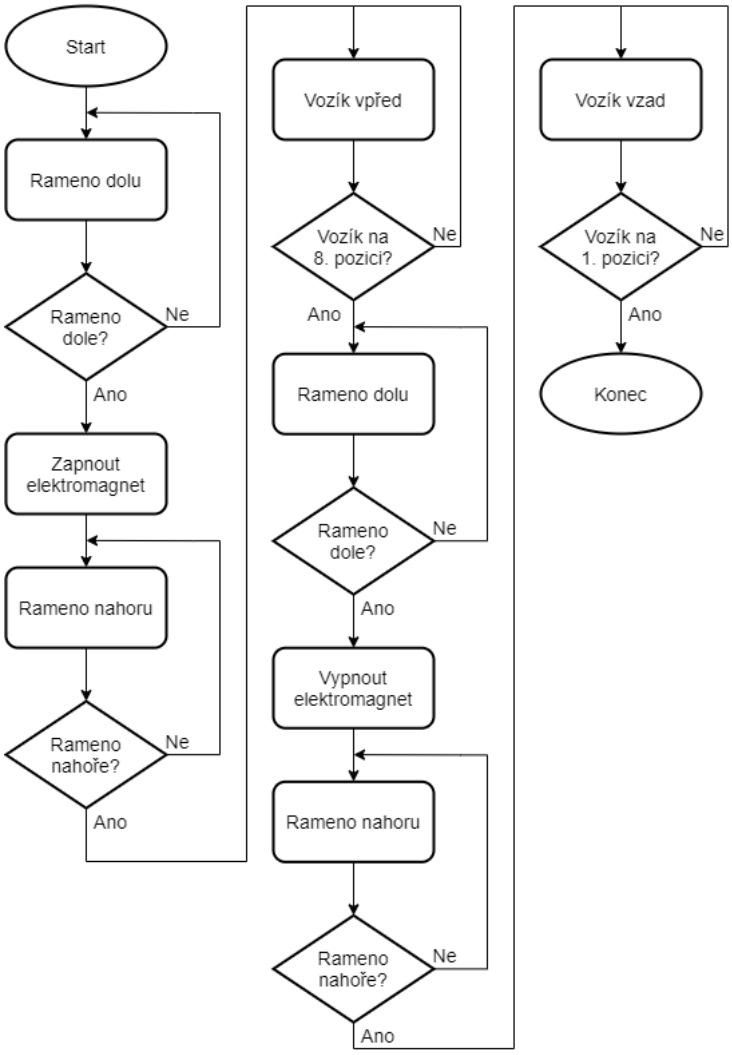


Obr. 3-17: Vývojový diagram subrutiny ‚Inicializace'

Subrutina ‚Přesun A' je zodpovědná za přesun bedýnky „tam", tj. uchopí bedýnku z první pozice a přesune ji na poslední pozici.

Vývojový diagram pro subrutinu ‚Přesun B' netřeba uvádět, protože funkce je podobná s ‚Přesun A' až na to, že vozík nejprve sjede na poslední 8. pozici uložení bedýnky a bedýnku přesune na pozici první. Tedy v diagramu dojde k prohození první a poslední pozice a na počátek přibude přesun na poslední pozici.

To, který přesun se vykoná, je zřejmé z vývojového diagramu rutiny ‚Jádro', která se zeptá, zda je první pozice obsazena bedýnkou. Pokud je, vykoná se přesun „tam", pokud ne, vykoná se přesun „zpět".



Obr. 3-18: Vývojový diagram subrutiny ‚Přesun A'

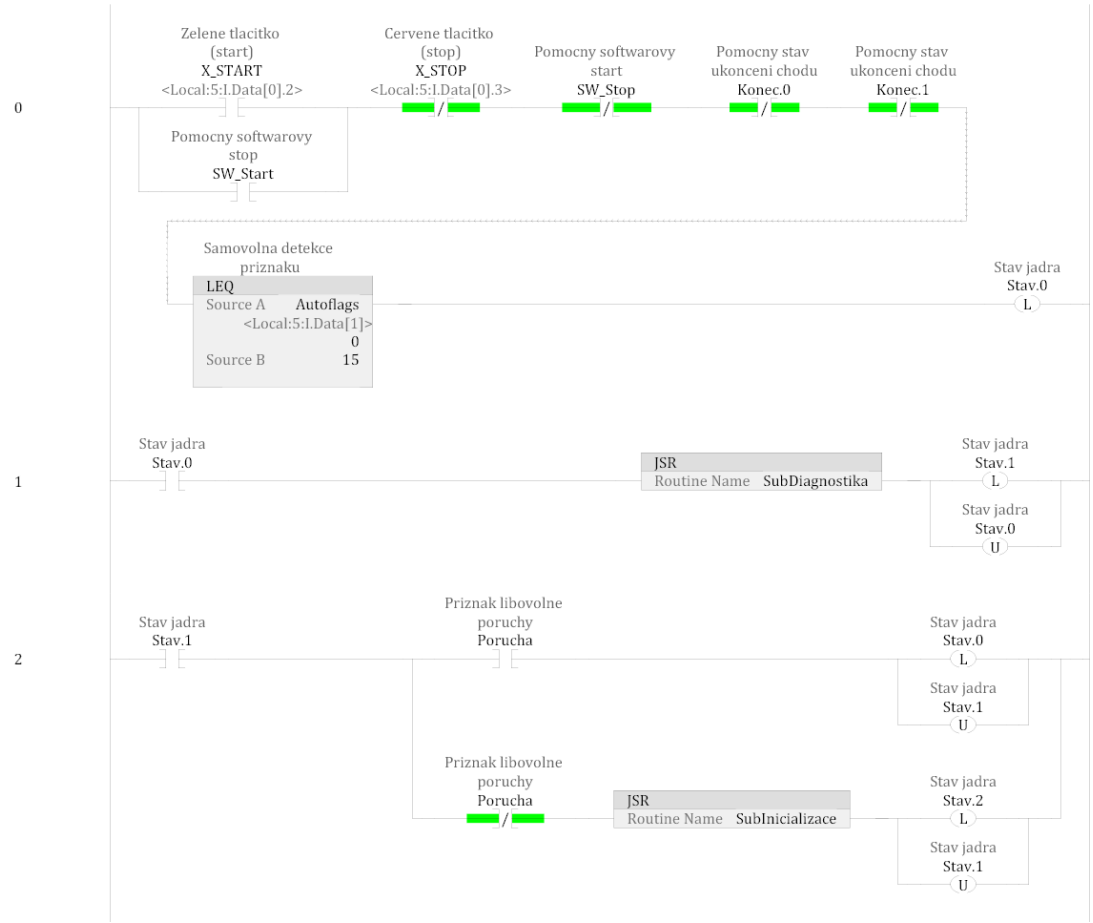
3.4.3 Konstrukce logiky

Po seznámení s vývojovými diagramy lze přistoupit k ukázce programu napsaného v jazyce ladder diagram ve *Studio 5000*.

Budou zde vynechány subrutiny inicializace a přesun tam i zpět, lze je nalézt až v příloze č. 4. Jejich detailní popis není přímo předmětem této práce, přesto nebudou vynechány z přílohy, protože s jejich existencí program funguje jako logický a funkční celek, netýká se pouze diagnostiky.

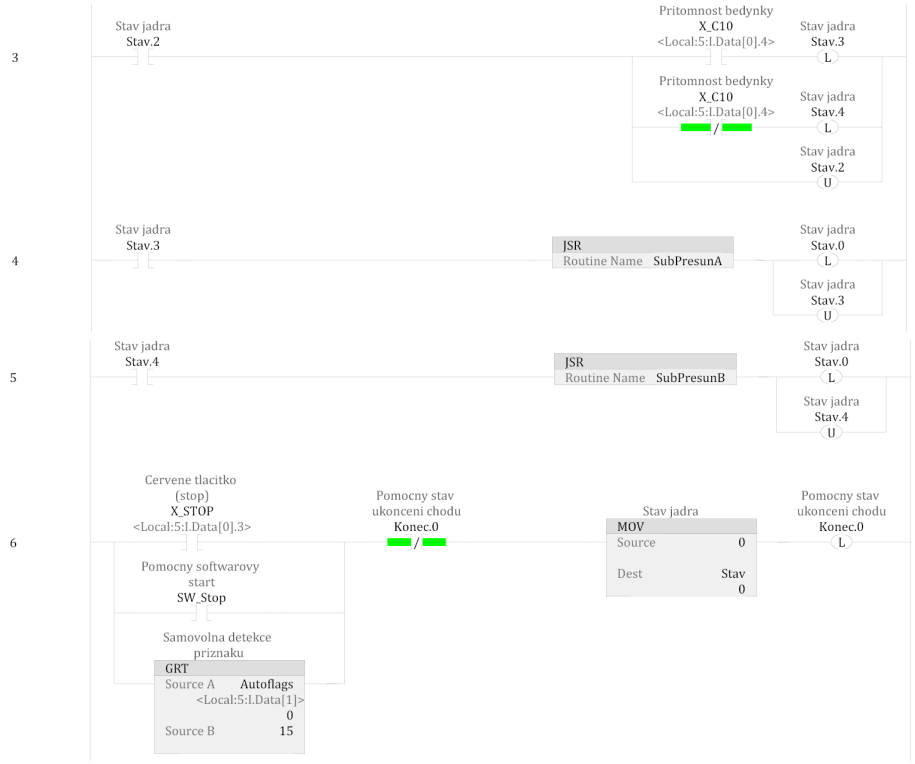
Řízení příček programu je zajištěno pomocí postupně spínaných a rozpínaných stavů dané rutiny. Tento přístup se zdál jako nejbezpečnější a snadno uchopitelný, umožňuje také jednodušší zpětné zásahy, když je potřeba upravit určité úseky. Byla snaha dodržet sled stavů shora dolů, aby uživatel nemusel přeskakovat v kódu.

První popsanou rutinou bude ta hlavní, tzv. ‚Jádro'. Věnují se jí obrázky 3-19 až 3-22 s bližším popisem chováním jednotlivých příček.



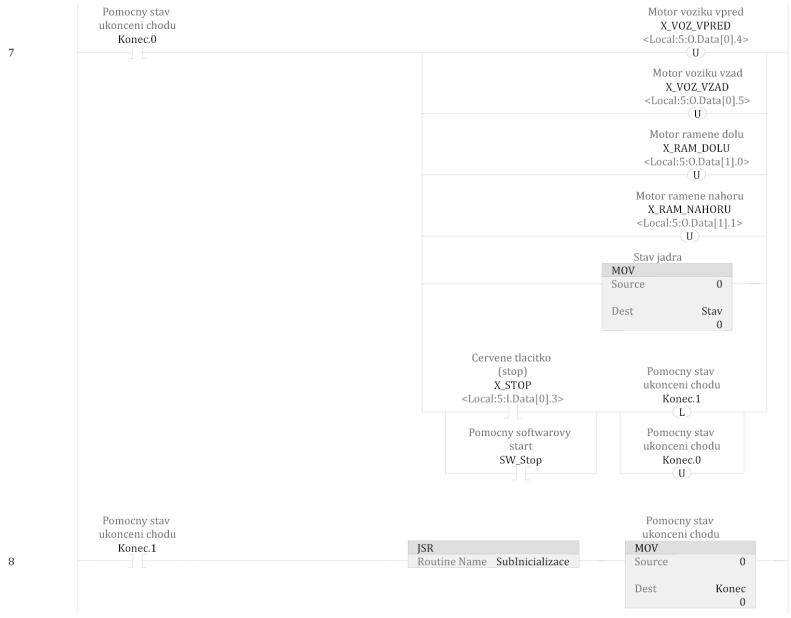
Obr. 3-19: Hlavní rutina ‚Jádro' 1/4

- **0. příčka** vyhodnocuje start, jak softwarový, tak hardwarový, a může být blokována poruchou nebo stopem.
- **1. příčka** spustí subrutinu diagnostiky.
- **2. příčka** spustí subrutinu inicializace v případě, že nenastala porucha. V případě poruchy se opakovaně volá diagnostika, aby měl uživatel nejnovější informace o stavu dopravníku.
- **3. příčka** vyhodnotí, zda je na prvním uložení přítomna bedýnka.
- **4. příčka** se sepne, pokud je bedýnka přítomna, a spustí přesun „tam".
- **5. příčka** se sepne, pokud bedýnka není přítomna, a spustí se přesun „zpět".
- **6. příčka** by se dala považovat za nový paralelní proces ke startu, jak je i vidět z vývojového diagramu, a detekuje, zda byl proveden stop, nebo nastala porucha.



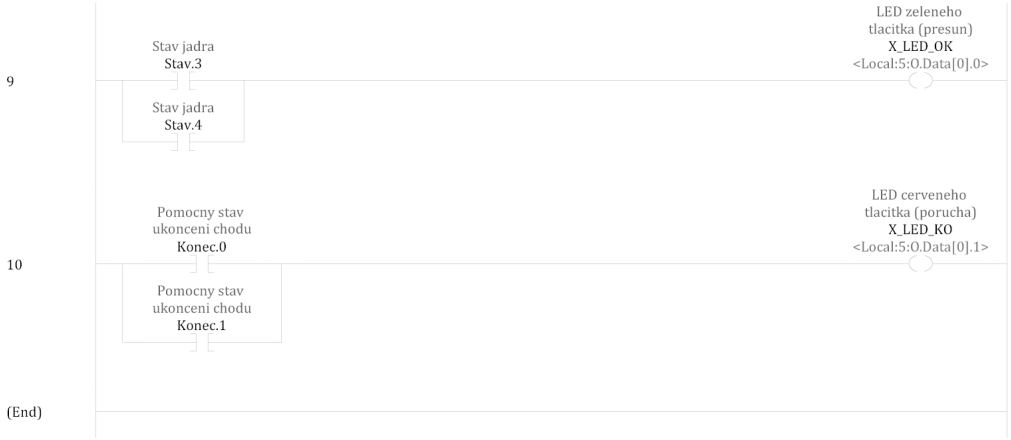
Obr. 3-20: Hlavní rutina ‚Jádro' 2/4

- **7. příčka** zastaví oba motory v libovolném směru a obnoví stav hlavní rutiny. Zároveň čeká na druhý stisk stop, který tento stav uvolní.
- **8. příčka** spustí subrutinu inicializace dopravníku a vynuluje stav ukončení.



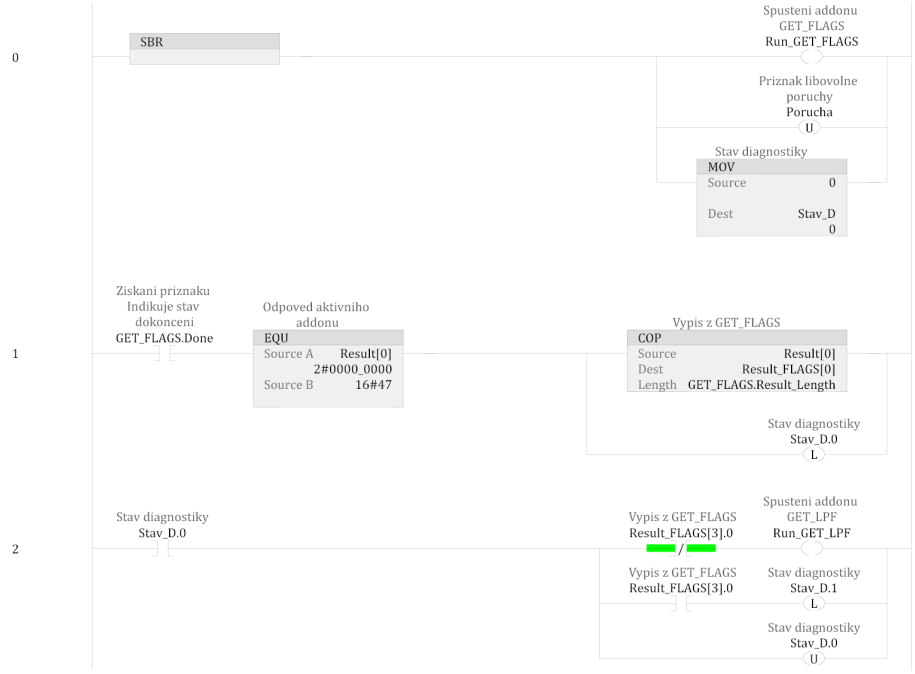
Obr. 3-21: Hlavní rutina ‚Jádro' 3/4

- **9. příčka** rozsvítí zelenou žárovku v případě, že je dopravník ve stavu přesunu „tam“ nebo „zpět“. Je tedy v regulérním chodu.
- **10. příčka** rozsvítí červenou žárovku v případě, že dopravník je ve stavu ukončení svého chodu, což může být způsobeno i poruchou.



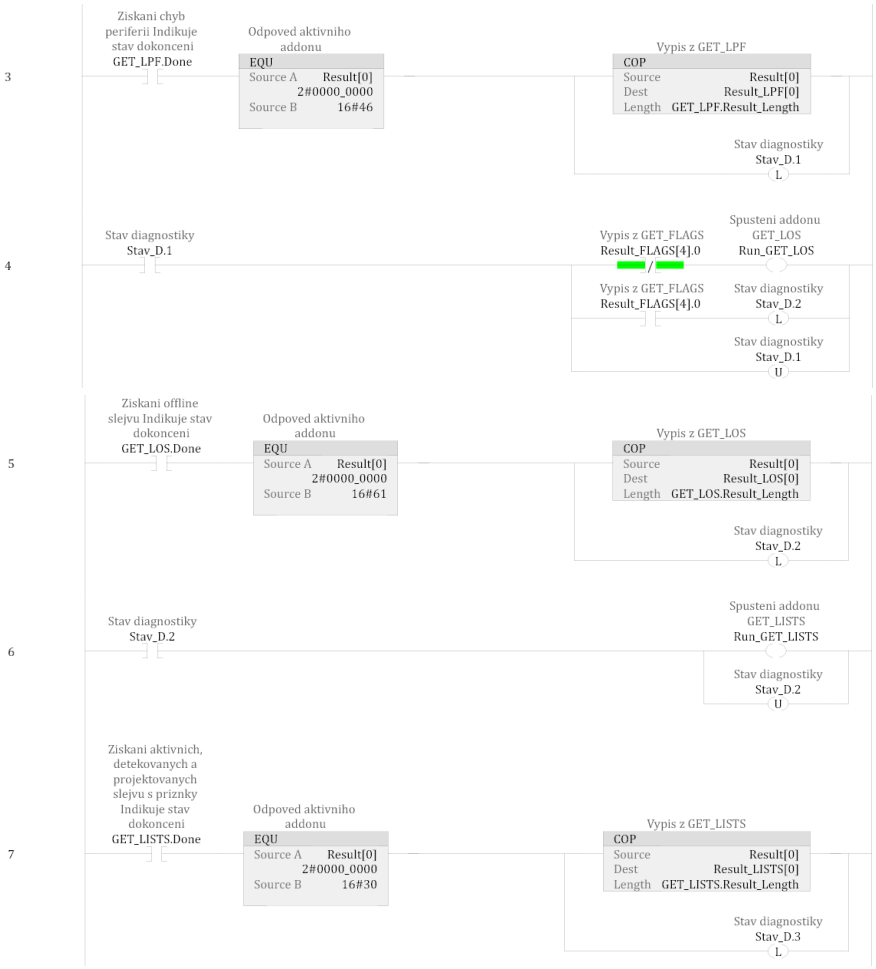
Obr. 3-22: Hlavní rutina ‚Jádro‘ 4/4

Druhou a zde poslední popsanou rutinou bude ‚Diagnostika‘. Věnují se jí obrázky 3-23 až 3-26 s bližším popisem chování jednotlivých příček. Rutina je stěžejní pro celou bakalářskou práci, protože ukazuje praktické použití vytvořených Add-On instrukcí, jak je spouštět, reagovat na jejich dokončení a ukládat z nich informace.



Obr. 3-23: Subrutina ‚Diagnostika‘ 1/3

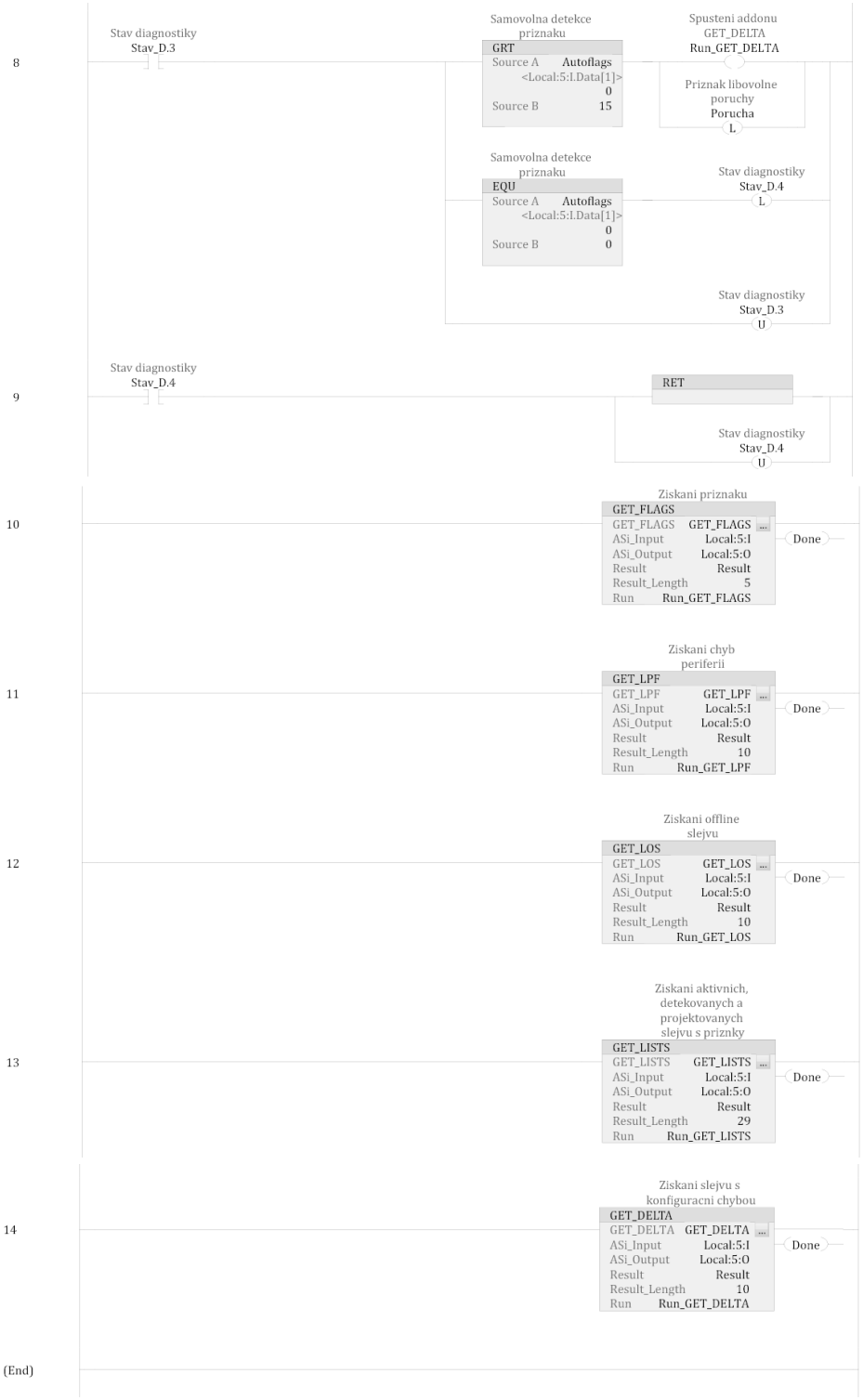
- **0. příčka** inicializuje rutinu a spustí Add-On pro získání příznaků GET_FLAGS.
- **1. příčka** ověří shodu odpovědi a uloží výsledek z GET_FLAGS o dané délce.
- **2. příčka** vyhodnotí, zda nastala porucha periferie, a podle toho případně spustí Add-On GET_LPF získávající seznam chyb periferií.
- **3. příčka** ověří shodu odpovědi a uloží výsledek z GET_LPF o dané délce.
- **4. příčka** vyhodnotí, zda nastala porucha konfigurace a podle toho případně spustí Add-On GET_LOS získávající seznam offline slejvů.
- **5. příčka** ověří shodu odpovědi a uloží výsledek z GET_LOS o dané délce.



Obr. 3-24: Subrutina ‚Diagnostika' 2/3

- **6. příčka** spustí Add-On instrukci pro získání seznamu aktivních, detekovaných a projektovaných slejvů GET_LISTS.
- **7. příčka** ověří shodu odpovědi a uloží výsledek z GET_LISTS o dané délce.
- **8. příčka** podle existence libovolného příznaku případně spustí Add-On GET_DELTA, který porovná seznamy získávající například Add-On GET_LISTS a vyhodnotí neshody.

- **9. příčka** slouží k návratu ze subrutiny.
- **10. až 14. příčka** obsahuje vynesené používané Add-On instrukce řízené z vyšších příček.



Obr. 3-25: Subrutina ‚Diagnostika' 3/3

3.5 Provedení rekonstrukce

V kapitole ‚2.2.2 Plánované zásahy do konstrukce' byly zmíněny hlavní nedostatky, s kterými se dopravník potýkal. Ve spolupráci s fakultou jsem z objednaných potřebných materiálů nechal vyhotovit nové uchycení motoru ramene. Vozíku jsem vyvrtal nové otvory pro uchycení některých snímačů a zvětšil průchody pro hřeben.

Protože je nyní hřeben mnohem silnější, nabídlo se šikovné uchycení elektromagnetu. Do hřebene byl navrtán vrut, do vrchu elektromagnetu vyvrtána díra, která se vyplnila tepelně odolným lepidlem. Vrut s odštíplou hlavičkou na konci hřebene se následně zapustil do díry v elektromagnetu, kdy po zatuhnutí vznikl velmi odolný spoj.

Po provedení rekonstrukce jsem úspěšně otestoval, zda všechny snímače a akční členy reagují a nacházejí se na stejných adresách v polích vstupů a výstupů ASi mastera. Nechtěl jsem konfiguraci nijak měnit, aby byla zajištěna zpětná kompatibilita s dříve vytvořenými programy pro tento dopravník.

4 ZÁVĚR

Čtenář by měl být úspěšně seznámen se základními principy sítě AS-Interface a zhodnotit její přednosti a omezení při průmyslovém nasazení. V práci bylo vytvořeno 36 Add-On instrukcí pro tuto síť s tím, že v demonstrační úloze pro model dopravníku bylo využito 5 vhodných diagnostických Add-On instrukcí sledujících průběžný stav dopravníku.

Koncept instrukcí byl úspěšně otestován, zda správně komunikuje s oběma ASi mastery a zda jsou schopny dosáhnout výměny dat po síti. Na základě toho jsem přistoupil k hromadné tvorbě instrukcí. Dbal jsem na co největší jednoduchost logiky odstupňováním do stavů, aby případné zpětné zásahy mohly probíhat lokálně a nemuselo docházet k rozsáhlým úpravám kombinačních podmínek kódu. Programátor zasahuje pouze do oblasti stavu.

Realizace instrukcí pro mastera BWU1416 byla jednodušší díky dříve uložené konfiguraci z předchozích prací na dopravníku. V případě BWU1488 bylo třeba fyzicky provést adresaci slejvů na masterovi po připojení dopravníku a tuto konfiguraci uložit do jeho paměti. Následný import modulu mastera ve vývojovém prostředí byl o něco komplikovanější, protože způsoboval chybnou konverzi datového typu, a čtená data tím pádem master odmítal. Přistoupil jsem k nahrání opravené konfigurace získané od podpory Bihl+Wiedemann, která umožnila korektní čtení oficiálního importovaného modulu. Konfigurační soubor modulu BWU1488 je součástí přílohy práce.

Vybrané diagnostické Add-On instrukce byly implementovány do demonstrační úlohy, jejíž nedílnou součástí je právě diagnostika. Úloha dále umožňuje základní přesouvání bedýnek, inicializaci a bezpečný start/stop.

Dopravník byl podroben rekonstrukci vozíku, v kterém byly vyměněny původní sešlé převody motoru ramene za hrubší, silnější, a s tím bylo vytvořeno i pevnější uchycení elektromagnetu k ramenu vozíku. Způsob zapojení snímačů a akčních členů ve slejv modulech zůstalo zachováno, aby byla zajištěna zpětná kompatibilita s předchozími programovými pracemi na dopravníku.

Všech 72 realizovaných instrukcí (36 pro každého mastera) bylo vyexportováno a lze je dál využívat v jiných ASi aplikacích, ne pouze pro dopravník. V přiložených souborech lze dále najít i program s demonstrační úlohou a vygenerovaný report.

Literatura

[1] BECKER, Rolf. *Automatizace je jednoduchá – s AS-INTERFACE*. Překlad: 2012 © AS-interface Česká republika. Frankfurt: Heinrich Druck + Medien, 2008, 96 s.

[2] *Logo AS-Interface*. In: AS-Interface [online]. Gelnhausen: AS-International Association, [2018] [cit. 2019-12-19]. Dostupné z: https://www.as-interface.net/typo3conf/ext/asinterface/Resources/Public/img/logo.jpg

[3] *AS-interface Česká republika: Základní informace o sběrnici AS-I* [online]. Brno: FEKT VUT v Brně [cit. 2019-12-19]. Dostupné z: http://www.as-interface.cz/AS-i_zaklad.html

[4] *ASi-5* [online]. Gelnhausen: AS-International Association [cit. 2019-12-19]. Dostupné z: https://www.asi-5.net/anwendernutzen/

[5] BECKER, Rolf, ref. 1, s. 39.

[6] BECKER, Rolf. *AS-Interface Řešení pro automatizaci: příručka, technika, funkce, aplikace*. AS-International Association, 2004, 184 s. ISBN 80-214-2958-5.

[7] ČSN 33 2000-4-41 ed. 3: *Elektrické instalace nízkého napětí – Část 4-41: Ochranná opatření pro zajištění bezpečnosti – Ochrana před úrazem elektrickým proudem*. Ed. 3. Technické Normy, 2018, 36 s.

[8] FIEDLER, Petr. In: *Seznámení s technologií průmyslové sběrnice AS-Interface*: prezentace, s. 8 [online]. Brno, 2010 [cit. 2019-12-19]. Dostupné z: http://www.as-interface.cz/Seminar/ASIV3_0.pdf

[9] BECKER, Rolf, ref. 1, s. 67.

[10] HOLOMEK, David. *Model překladače*. Brno, 2013. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Ing. Radek Štohl, Ph.D.

[11] SIEMENS. *Product data sheet 3RK1400-1MQ03-0AA4* [online]. In: 2013 [cit. 2019-12-19]. Dostupné z: https://datasheet.octopart.com/3RK1400-1MQ03-0AA4-Siemens-datasheet-14371597.pdf

[12] ifm electronic. *AC2086: AS-Interface illuminated pushbutton module* [online]. In: Essen, 2015, 16. 12. 2015 [cit. 2019-12-19]. Dostupné z: https://www.ifm.com/de/en/product/AC2086

[13] ifm electronic. *AC5214: AS-Interface ClassicLine module with quick mounting technology* [online]. In: Essen, 2013, 3. 7. 2013 [cit. 2019-12-19]. Dostupné z: https://www.ifm.com/de/en/product/AC5214

[14] ifm electronic. *AC5210: AS-Interface ClassicLine module with quick mounting technology* [online]. In: Essen, 2011, 21. 12. 2011 [cit. 2019-12-19]. Dostupné z: https://www.ifm.com/de/en/product/AC5210

[15] ifm electronic. AC5222: *AS-Interface ClassicLine module with quick mounting technology* [online]. In: Essen, 2012, 18. 7. 2012 [cit. 2019-12-19]. Dostupné z: https://www.ifm.com/de/en/product/AC5222
[16] Bihl+Wiedemann. *BWU1416* [online]. In: Mannheim [cit. 2019-12-19]. Dostupné z: https://www.bihl-wiedemann.de/fileadmin/_processed_/3/a/csm_1416_1610_f8a5663486.jpg
[17] *CompactLogix 1769-L33ERM* [cit. 2019-12-19]. Dostupné z: https://www.nexinstrument.com/assets/images/1769-L33ERM.jpg
[18] Bihl+Wiedemann. *BWU1488* [online]. In: Mannheim [cit. 2019-12-19]. Dostupné z: https://www.bihl-wiedemann.de/fileadmin/_processed_/f/e/csm_1488_1611_44acf95208.jpg
[19] *GuardLogix 1756-L72S* [cit. 2019-12-19]. Dostupné z: https://5.imimg.com/data5/VL/CC/UW/SELLER-5457274/1756-l72s-500x500.jpg
[20] *ASi napájecí zdroj AC1224* [cit. 2019-12-19]. Dostupné z: http://eshop.technoline.cz/shop_pictures/full/AC1224.gif
[21] BKE. *JS-55-240/DIN* [online]. In: Hrušovany u Brna [cit. 2019-12-19]. Dostupné z: https://www.bke.cz/sites/default/files/styles/large/public/perm/product/35.png
[22] Rockwell Automation Publication. *Logix 5000 Controllers Add On Instructions* [online]. In: 2018 [cit. 2019-12-19]. Dostupné z: https://literature.rockwellautomation.com/idc/groups/literature/documents/pm/1756-pm010_-en-p.pdf
[23] Pepperl+Fuchs. AS-i 3.0 Command Interface [online]. In: Germany, 2013 [cit. 2019-12-19]. Dostupné z: https://files.pepperl-fuchs.com/webcat/navi/productInfo/doct/tdoct1484a_eng.pdf
[24] BECKER, Rolf, ref. 1, s. 51.
[25] BECKER, Rolf, ref. 1, s. 52.
[26] Bihl+Wiedemann. *AS-Interface Master/Scanner for ALLEN-BRDALEY User Manual* [online]. In: Mannheim, 2006 [cit. 2019-12-19]. Dostupné z: https://www.ifm.com/img/ControlLogixBPManual.pdf

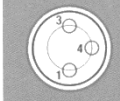
Seznam symbolů, veličin a zkratek

ASi	–	Actuator/Sensor Interface
PLC	–	Programmable Logic Controller
I/O	–	Input/Output
D/A	–	Digitální/Analogový
LAD	–	Ladder Diagram
ST	–	Strukturovaný text
VDC	–	Stejnosměrné napětí ve voltech

Seznam příloh

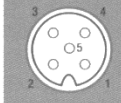
1. Konektory M8 a M12
2. Datová pole vstupů a výstupů AS-Interface
3. Výpis ukázkového programu
4. Výpis vytvořených instrukcí a jejich export
5. EDS soubor pro BWU1488

Příloha 1: Konektory M8 a M12

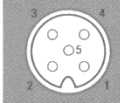


AS-Interface	Pomocné napájení	Binární vstup	Binární výstup (polovodič)	Binární výstup (relé)
1 – ASi +	1 – AUX +	1 – U+(24 V DC)	1 – U+(24 V DC)	1 – střed
3 – ASi -	3 – AUX -	3 – 0 V	3 – 0 V	3 –
4 –	4 –	4 – spínací vstup	4 – spínací výstup	4 – spínací kontakt (NO)

Obr. P1-1: Standardizované schéma připojení pro konektor M8 [24]



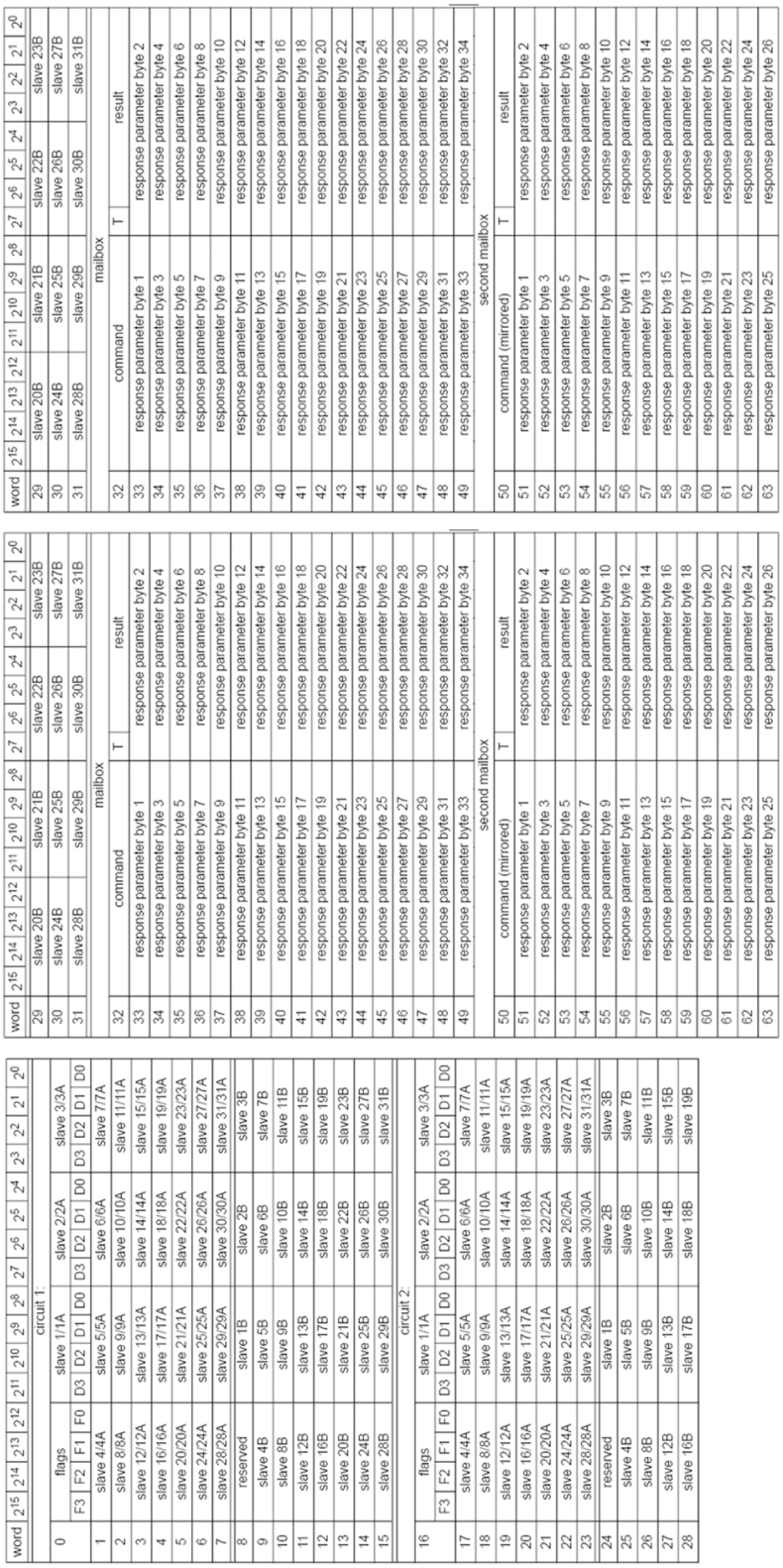
AS-Interface	Pomocné napájení	AS-i a pomocné napájení	Binární vstup	Binární výstup (polovodič)
1 – ASi +	1 – AUX +	1 – ASi +	1 – U+(24 V DC)	1 – U+(24 V DC)
2 –	2 –	2 – AUX -	2 – spínací vstup 2	2 –
3 – ASi -	3 – AUX -	3 – ASi -	3 – 0 V	3 – 0 V
4 –	4 –	4 – AUX +	4 – spínací vstup 1	4 – spínací výstup
5 –	5 – (GND)	5 – (GND)	5 – (GND)	5 – (GND)



Dvojitý binární vystup	Binární výstup (relé)	Analogový vstup	Vstup pro snímač Pt100	Analogový výstup
1 – U+(24 V DC)	1 – střed	1 – U +	1 – Iconst +	1 – Out
2 – spínací vstup 2	2 – rozpínací kontakt (NC)	2 – In +	2 – In +	2 –
3 – 0 V	3 –	3 – 0 V	3 – Iconst -	3 – 0 V
4 – spínací vstup 1	4 – spínací kontakt (NO)	4 – In -	4 – In -	4 –
5 – (GND)	5 – (GND)	5 – (GND)	5 – (GND)	5 – (GND)

Obr. P1-2: Standardizované schéma připojení pro konektor M8 [25]

Příloha 2: Datová pole vstupů a výstupů AS-Interface



word	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	circuit 1:															
0	flags				slave 1/1A				slave 2/2A				slave 3/3A			
	F3	F2	F1	F0	D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0
1	slave 4/4A				slave 5/5A				slave 6/6A				slave 7/7A			
2	slave 8/8A				slave 9/9A				slave 10/10A				slave 11/11A			
3	slave 12/12A				slave 13/13A				slave 14/14A				slave 15/15A			
4	slave 16/16A				slave 17/17A				slave 18/18A				slave 19/19A			
5	slave 20/20A				slave 21/21A				slave 22/22A				slave 23/23A			
6	slave 24/24A				slave 25/25A				slave 26/26A				slave 27/27A			
7	slave 28/28A				slave 29/29A				slave 30/30A				slave 31/31A			
8	reserved				slave 1B				slave 2B				slave 3B			
9	slave 4B				slave 5B				slave 6B				slave 7B			
10	slave 8B				slave 9B				slave 10B				slave 11B			
11	slave 12B				slave 13B				slave 14B				slave 15B			
12	slave 16B				slave 17B				slave 18B				slave 19B			
13	slave 20B				slave 21B				slave 22B				slave 23B			
14	slave 24B				slave 25B				slave 26B				slave 27B			
15	slave 28B				slave 29B				slave 30B				slave 31B			
	circuit 2:															
16	flags				slave 1/1A				slave 2/2A				slave 3/3A			
	F3	F2	F1	F0	D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0
17	slave 4/4A				slave 5/5A				slave 6/6A				slave 7/7A			
18	slave 8/8A				slave 9/9A				slave 10/10A				slave 11/11A			
19	slave 12/12A				slave 13/13A				slave 14/14A				slave 15/15A			
20	slave 16/16A				slave 17/17A				slave 18/18A				slave 19/19A			
21	slave 20/20A				slave 21/21A				slave 22/22A				slave 23/23A			
22	slave 24/24A				slave 25/25A				slave 26/26A				slave 27/27A			
23	slave 28/28A				slave 29/29A				slave 30/30A				slave 31/31A			
24	reserved				slave 1B				slave 2B				slave 3B			
25	slave 4B				slave 5B				slave 6B				slave 7B			
26	slave 8B				slave 9B				slave 10B				slave 11B			
27	slave 12B				slave 13B				slave 14B				slave 15B			
28	slave 16B				slave 17B				slave 18B				slave 19B			

word	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
29	slave 20B				slave 21B				slave 22B				slave 23B			
30	slave 24B				slave 25B				slave 26B				slave 27B			
31	slave 28B				slave 29B				slave 30B				slave 31B			
	mailbox															
32	command							T	result							
33	response parameter byte 1								response parameter byte 2							
34	response parameter byte 3								response parameter byte 4							
35	response parameter byte 5								response parameter byte 6							
36	response parameter byte 7								response parameter byte 8							
37	response parameter byte 9								response parameter byte 10							
38	response parameter byte 11								response parameter byte 12							
39	response parameter byte 13								response parameter byte 14							
40	response parameter byte 15								response parameter byte 16							
41	response parameter byte 17								response parameter byte 18							
42	response parameter byte 19								response parameter byte 20							
43	response parameter byte 21								response parameter byte 22							
44	response parameter byte 23								response parameter byte 24							
45	response parameter byte 25								response parameter byte 26							
46	response parameter byte 27								response parameter byte 28							
47	response parameter byte 29								response parameter byte 30							
48	response parameter byte 31								response parameter byte 32							
49	response parameter byte 33								response parameter byte 34							
	second mailbox															
50	command (mirrored)							T	result							
51	response parameter byte 1								response parameter byte 2							
52	response parameter byte 3								response parameter byte 4							
53	response parameter byte 5								response parameter byte 6							
54	response parameter byte 7								response parameter byte 8							
55	response parameter byte 9								response parameter byte 10							
56	response parameter byte 11								response parameter byte 12							
57	response parameter byte 13								response parameter byte 14							
58	response parameter byte 15								response parameter byte 16							
59	response parameter byte 17								response parameter byte 18							
60	response parameter byte 19								response parameter byte 20							
61	response parameter byte 21								response parameter byte 22							
62	response parameter byte 23								response parameter byte 24							
63	response parameter byte 25								response parameter byte 26							

Obr. P2-1: Pole vstupních dat [26]

word	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
0	flags				slave 1/1A				slave 2/2A				slave 3/3A			
	F3	F2	F1	F0	D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0
circuit 1:																
1	slave 4/4A				slave 5/5A				slave 6/6A				slave 7/7A			
2	slave 8/8A				slave 9/9A				slave 10/10A				slave 11/11A			
3	slave 12/12A				slave 13/13A				slave 14/14A				slave 15/15A			
4	slave 16/16A				slave 17/17A				slave 18/18A				slave 19/19A			
5	slave 20/20A				slave 21/21A				slave 22/22A				slave 23/23A			
6	slave 24/24A				slave 25/25A				slave 26/26A				slave 27/27A			
7	slave 28/28A				slave 29/29A				slave 30/30A				slave 31/31A			
8	reserved				slave 1B				slave 2B				slave 3B			
9	slave 4B				slave 5B				slave 6B				slave 7B			
10	slave 8B				slave 9B				slave 10B				slave 11B			
11	slave 12B				slave 13B				slave 14B				slave 15B			
12	slave 16B				slave 17B				slave 18B				slave 19B			
13	slave 20B				slave 21B				slave 22B				slave 23B			
14	slave 24B				slave 25B				slave 26B				slave 27B			
15	slave 28B				slave 29B				slave 30B				slave 31B			
circuit 2:																
16	flags				slave 1/1A				slave 2/2A				slave 3/3A			
	F3	F2	F1	F0	D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0
17	slave 4/4A				slave 5/5A				slave 6/6A				slave 7/7A			
18	slave 8/8A				slave 9/9A				slave 10/10A				slave 11/11A			
19	slave 12/12A				slave 13/13A				slave 14/14A				slave 15/15A			
20	slave 16/16A				slave 17/17A				slave 18/18A				slave 19/19A			
21	slave 20/20A				slave 21/21A				slave 22/22A				slave 23/23A			
22	slave 24/24A				slave 25/25A				slave 26/26A				slave 27/27A			
23	slave 28/28A				slave 29/29A				slave 30/30A				slave 31/31A			
24	reserved				slave 1B				slave 2B				slave 3B			
25	slave 4B				slave 5B				slave 6B				slave 7B			
26	slave 8B				slave 9B				slave 10B				slave 11B			
27	slave 12B				slave 13B				slave 14B				slave 15B			
28	slave 16B				slave 17B				slave 18B				slave 19B			
29	slave 20B				slave 21B				slave 22B				slave 23B			

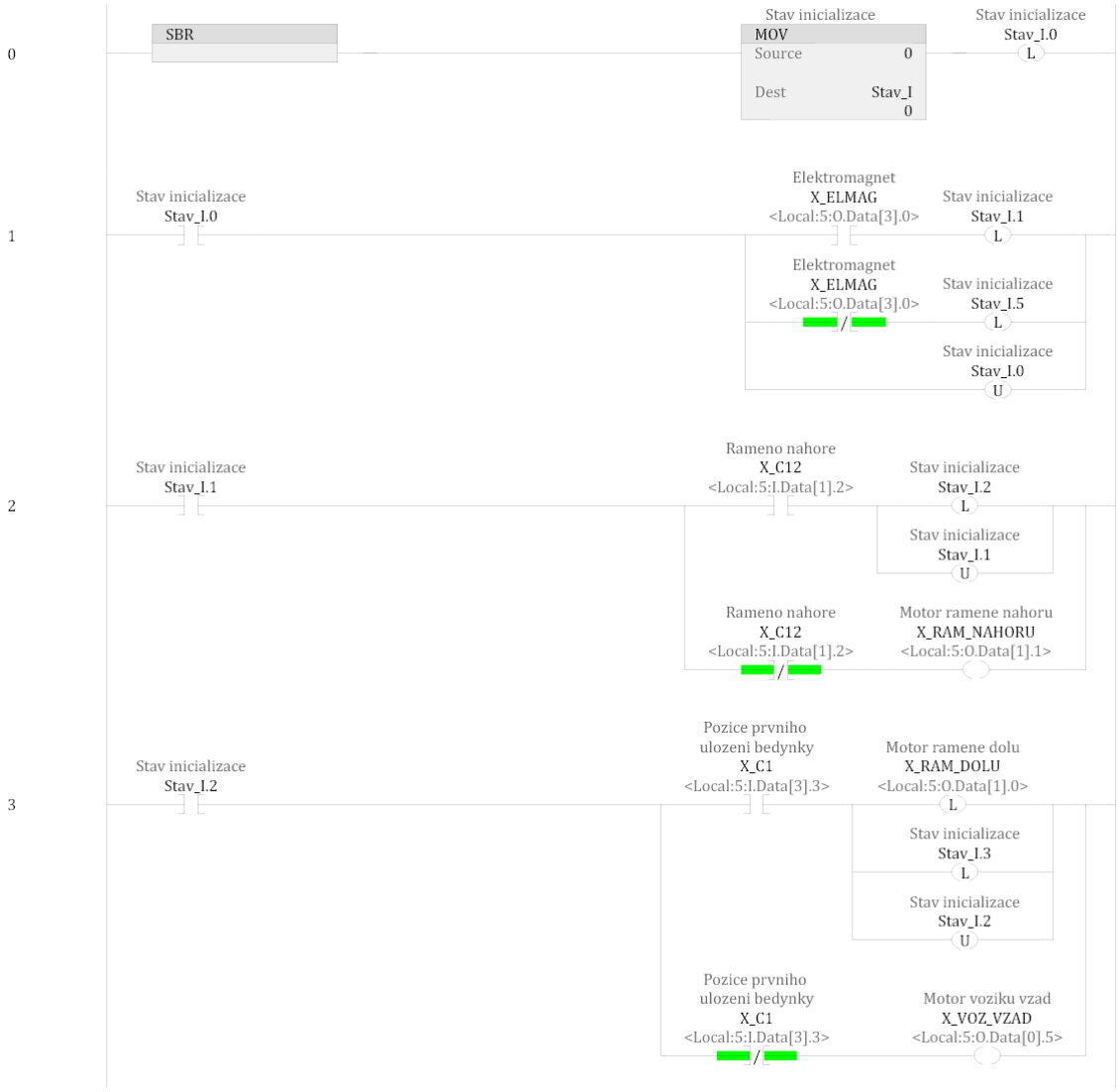
word	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
30	slave 24B				slave 25B				slave 26B				slave 27B			
31	slave 28B				slave 29B				slave 30B				slave 31B			
mailbox																
32	command											T	–	circuit		
33	request parameter byte 1												request parameter byte 2			
34	request parameter byte 3												request parameter byte 4			
35	request parameter byte 5												request parameter byte 6			
36	request parameter byte 7												request parameter byte 8			
37	request parameter byte 9												request parameter byte 10			
38	request parameter byte 11												request parameter byte 12			
39	request parameter byte 13												request parameter byte 14			
40	request parameter byte 15												request parameter byte 16			
41	request parameter byte 17												request parameter byte 18			
42	request parameter byte 19												request parameter byte 20			
43	request parameter byte 21												request parameter byte 22			
44	request parameter byte 23												request parameter byte 24			
45	request parameter byte 25												request parameter byte 26			
46	request parameter byte 27												request parameter byte 28			
47	request parameter byte 29												request parameter byte 30			
48	request parameter byte 31												request parameter byte 32			
49	request parameter byte 33												request parameter byte 34			
second mailbox																
50	command (mirrored)											T	result			
51	response parameter byte 1												response parameter byte 2			
52	response parameter byte 3												response parameter byte 4			
53	response parameter byte 5												response parameter byte 6			
54	response parameter byte 7												response parameter byte 8			
55	response parameter byte 9												response parameter byte 10			
56	response parameter byte 11												response parameter byte 12			
57	response parameter byte 13												response parameter byte 14			
58	response parameter byte 15												response parameter byte 16			
59	response parameter byte 17												response parameter byte 18			
60	response parameter byte 19												response parameter byte 20			
61	response parameter byte 21												response parameter byte 22			
62	response parameter byte 23												response parameter byte 24			
63	response parameter byte 25												response parameter byte 26			
64	response parameter byte 27												response parameter byte 28			

word	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
65	response parameter byte 29								response parameter byte 30							
66	response parameter byte 31								response parameter byte 32							
67	response parameter byte 33								response parameter byte 34							
analog output data circuit 1, slaves 10 … 31																
68	slave 10, channel 1															
69	slave 10, channel 2															
70	slave 10, channel 3															
71	slave 10, channel 4															
72	slave 11, channel 1															
73	slave 11, channel 2															
…	…															
150	slave 30, channel 3															
151	slave 30, channel 4															
152	slave 31, channel 1															
153	slave 31, channel 2															
154	slave 31, channel 3															
155	slave 31, channel 4															
analog output data circuit 2, slaves 10 … 31																
156	slave 10, channel 1															
157	slave 10, channel 2															
158	slave 10, channel 3															
159	slave 10, channel 4															
160	slave 11, channel 1															
161	slave 11, channel 2															
…	…															
238	slave 30, channel 3															
239	slave 30, channel 4															
240	slave 31, channel 1															
241	slave 31, channel 2															
242	slave 31, channel 3															
243	slave 31, channel 4															

Obr. P2-2: Pole výstupních dat [26]

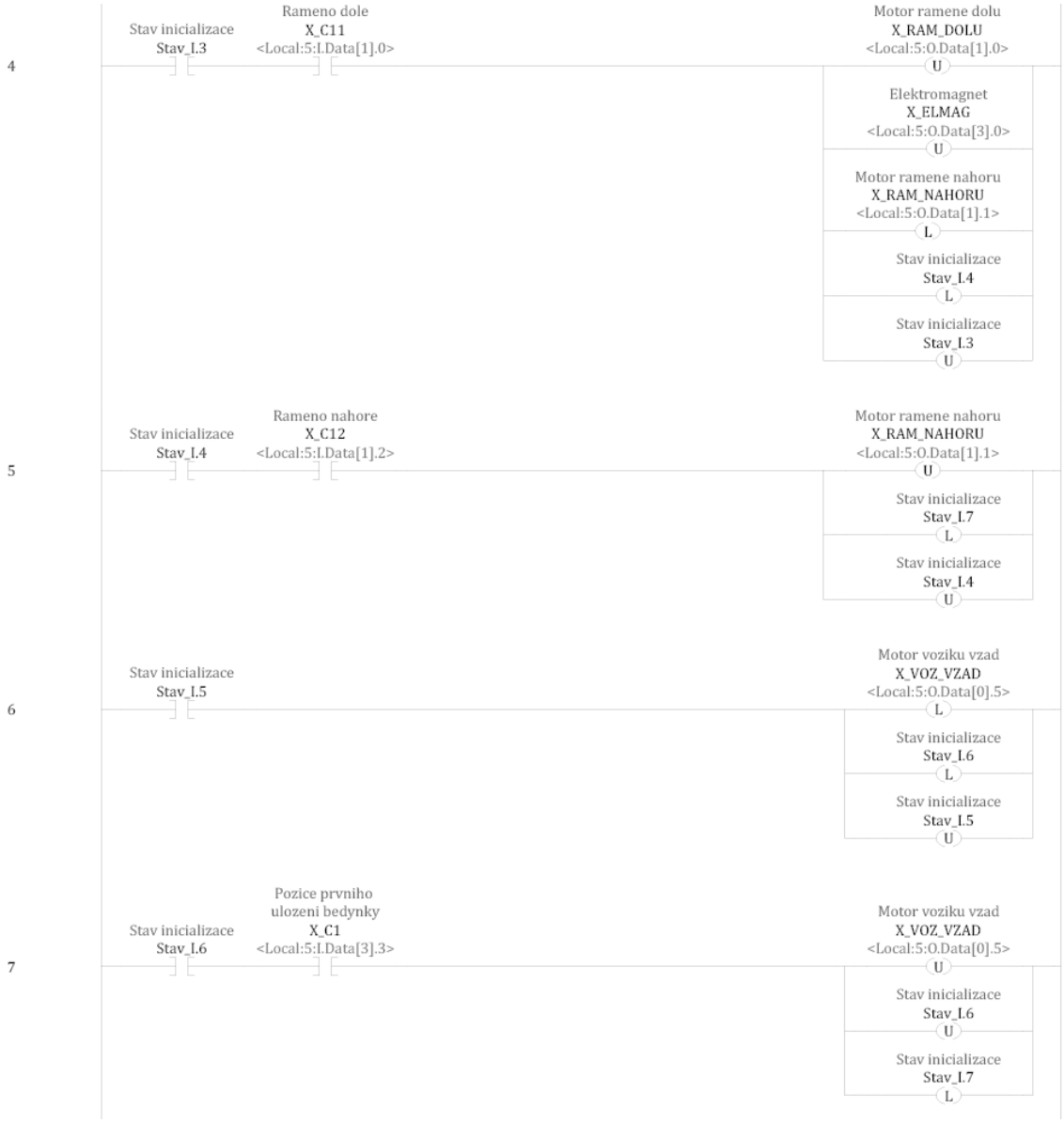
Příloha 3: Výpis ukázkového programu

Subrutina ‚Inicializace‘



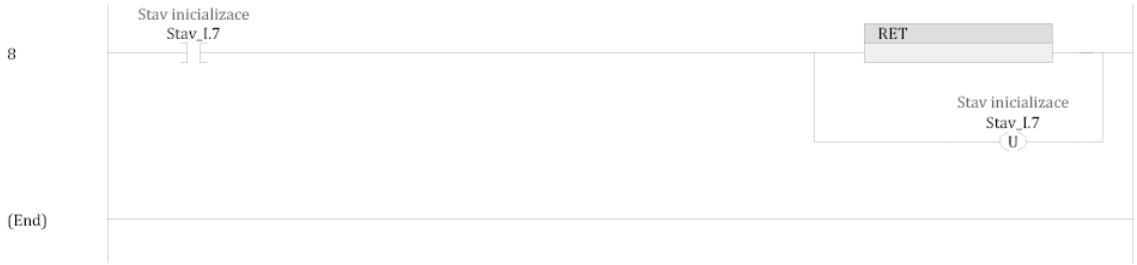
Obr. P3-1: Subrutina ‚Inicializace‘ 1/3

- **0. příčka** inicializuje subrutinu.
- **1. příčka** zjistí, zda je sepnutý elektromagnet. Pokud ano, pokračuje 2. příčkou.
- **2. příčka** zvedá rameno nahoru, dokud nenarazí na koncový snímač.
- **3. příčka** couvá s vozíkem, dokud nenarazí na 1. uložení bedýnky. Následně spustí rameno dolů.



Obr. P3-2: Subrutina ‚Inicializace' 2/3

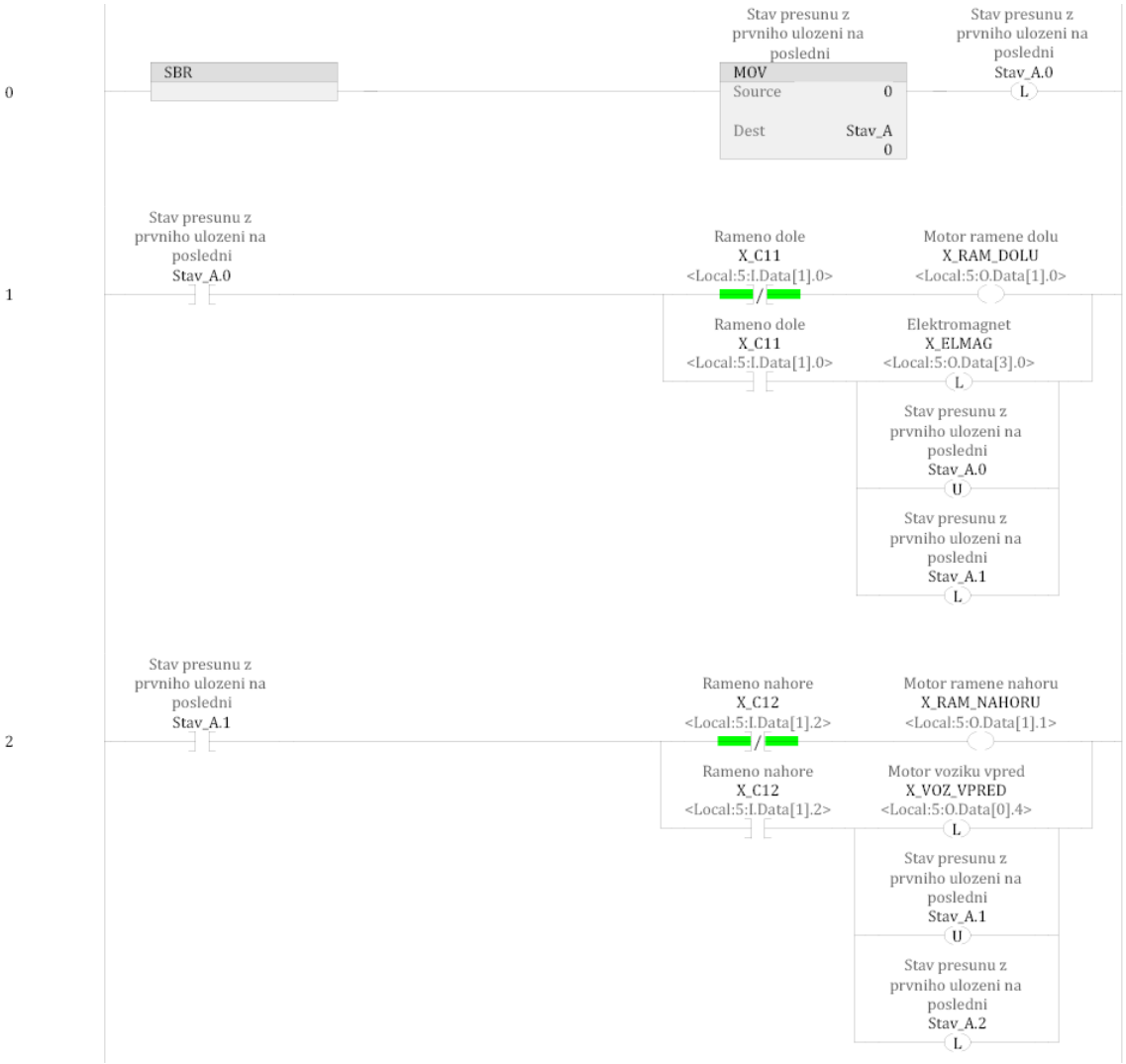
- **4. příčka** čeká na koncovou polohu ramena dole. Následně rameno zastaví, vypne elektromagnet a začne zvedat rameno nahoru.
- **5. příčka** čeká na koncovou polohu ramena nahoře. Poté vypne motor. Jedná se zároveň o příčku, která je počáteční, pokud není sepnutý elektromagnet.
- **6. příčka** pošle vozík vzad.
- **7. příčka** ověří, že je vozík na prvním uložení bedýnky a vypne motor vozíku. 6. příčka se může vůči 3. příčce zdát navíc, ovšem v případě vypnutého elektromagnetu zajistí první polohu uložení, a pokud tam vozík již je, je toto rychle ověřeno v 7. příčce a vozík se nestihne pohnout.



Obr. P3-3: Subrutina ‚Inicializace' 3/3

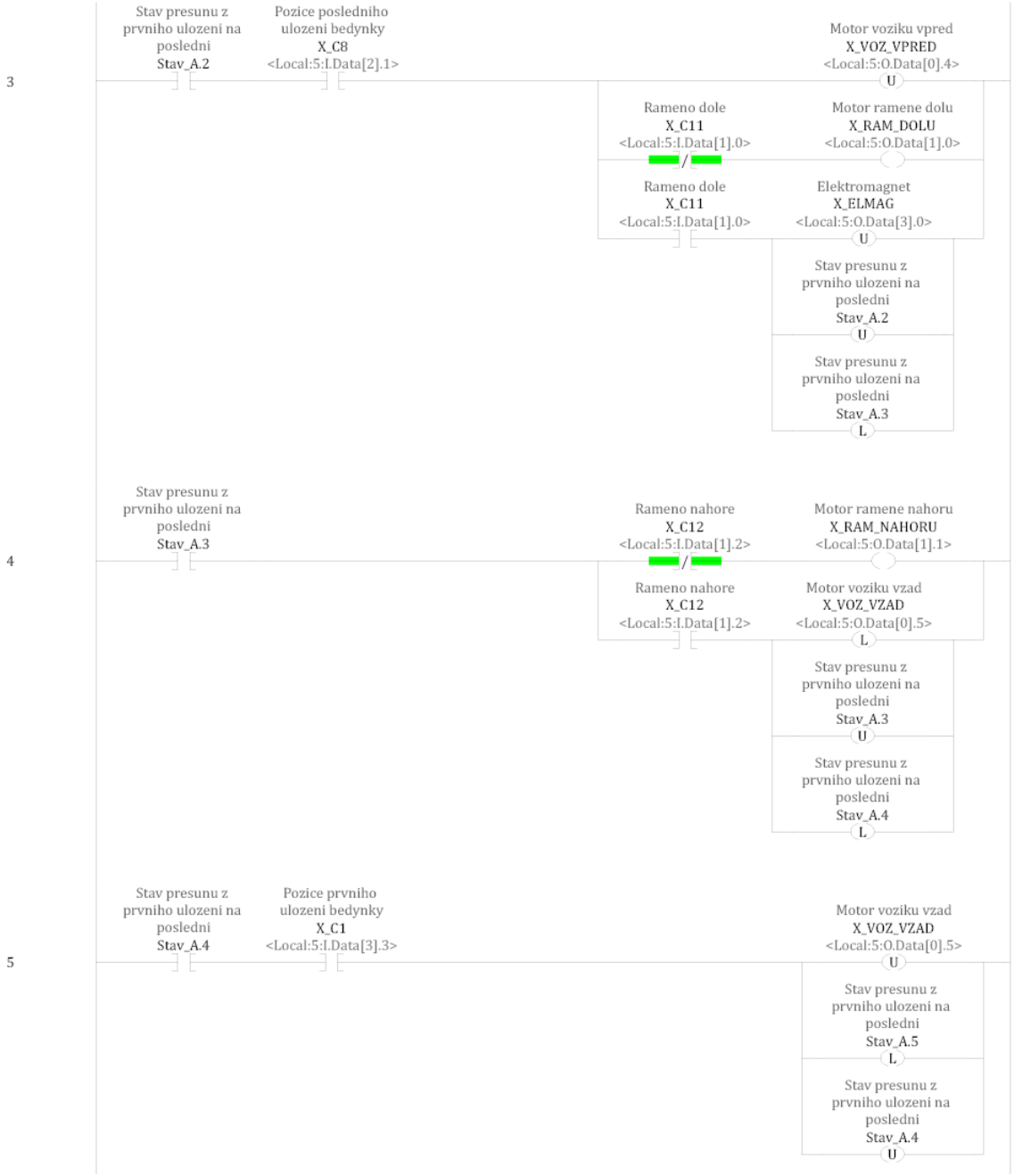
- **8. příčka** oznamuje návrat ze subrutiny.

Subrutina ‚Přesun A'



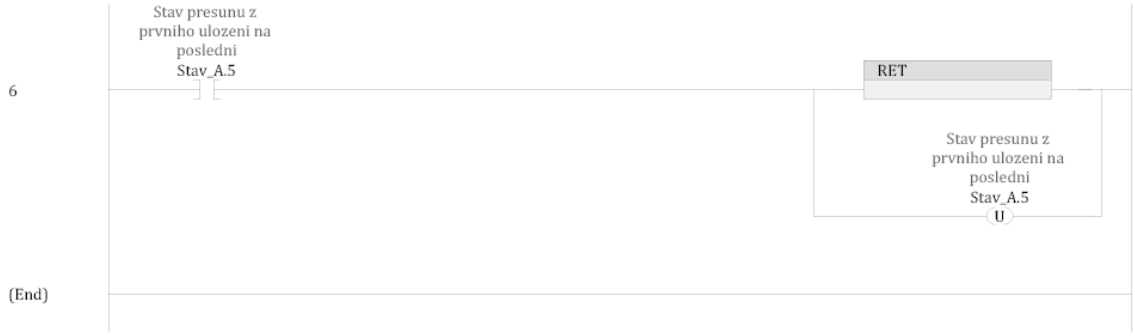
Obr. P3-4: Subrutina ‚Posun A' 1/3

- **0. příčka** inicializuje subrutinu.
- **1. příčka** pouští rameno dolů, dokud nenarazí na koncový snímač, kdy sepne elektromagnet pro uchopení bedýnky.
- **2. příčka** zvedá rameno nahoru, dokud nenarazí na koncový snímač, kde spustí vozík vpřed.



Obr. P3-5: Subrutina ‚Posun A' 2/3

- **3. příčka** vypne posun vozíku vpřed, jakmile narazí na 8. uložení bedýnky, a spustí rameno dolů, dokud nenarazí na koncový snímač, kdy následně vypne elektromagnet, aby uložil bedýnku.
- **4. příčka** zvedá rameno nahoru, dokud nenarazí na koncovou polohu, kdy spustí posun vozíku vzad.
- **5. příčka** čeká, dokud vozík nedojede na pozici prvního uložení bedýnky, poté motor vozíku vypne.



Obr. P3-6: Subrutina ‚Posun B' 3/3

- **6. příčka** oznamuje návrat ze subrutiny.

Subrutina ‚Přesun B'

Vychází ze subrutiny ‚Přesun A', kdy na počátku nejprve najede na poslední pozici a provádí přesun „zpět" na 1. uložení, nikoli „tam" na 8. uložení. Detail lze najít v příloze bakalářské práce s vygenerovaným reportem.

Příloha 4: Výpis vytvořených instrukcí a jejich export

Výpis obsahuje 36 Add-On instrukcí. Ke každé existuje popis funkce, podoba v jazyce ladder diagram, pole odpovědi a případně pole požadavku, pokud je vyžadována informace od uživatele.

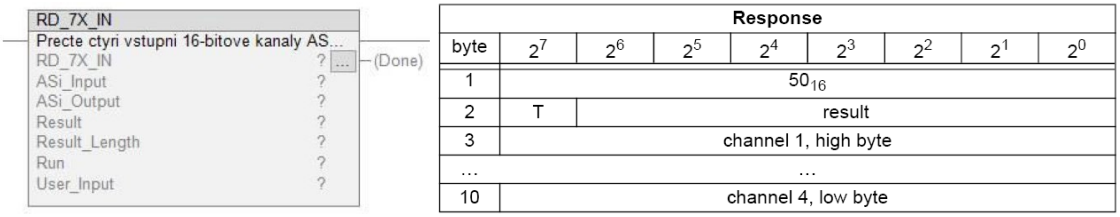
Nastavení parametrů je dle kapitoly *3.2.4 Implementace a další použití*. Data pro parametr *User_Input* doplňující pole požadavku, je specifikován pro danou Add-On instrukci.

Instrukce RD_7X_IN

Pomocí této instrukce lze přečíst čtyři vstupní 16bitové kanály ASi slejva dle profilu 7.3.

Tab. P4-1: Uživatelský vstup User_Input (SINT) RD_7X_IN

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		0	adresa slejva				



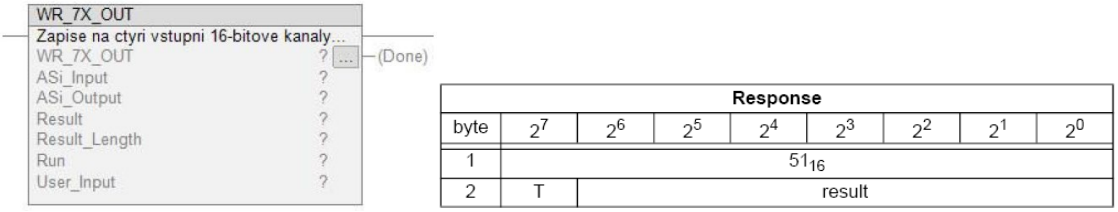
Obr. P4-1: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce WR_7X_OUT

Pomocí této instrukce lze zapsat na čtyři výstupní 16bitové kanály ASi slejva dle profilu 7.3.

Tab. P4-2: Uživatelský vstup User_Input (SINT[9]) WR_7X_OUT

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		0	adresa slejva				
2	kanál 1, horní byte							
	...							
9	kanál 4, dolní byte							



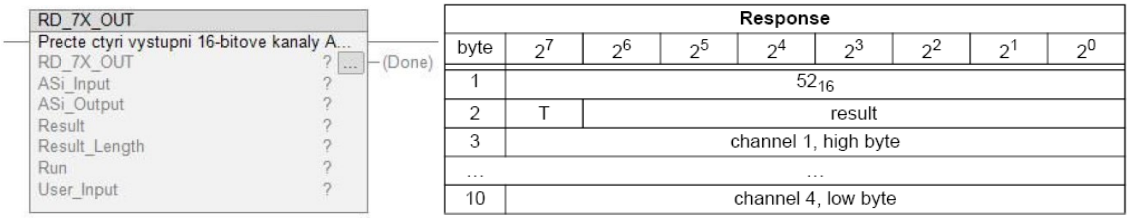
Obr. P4-2: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce RD_7X_OUT

Pomocí této instrukce lze přečíst čtyři výstupní 16bitové kanály ASi slejva dle profilu 7.3.

Tab. P4-3: Uživatelský vstup User_Input (SINT) RD_7X_OUT

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		0	adresa slejva				



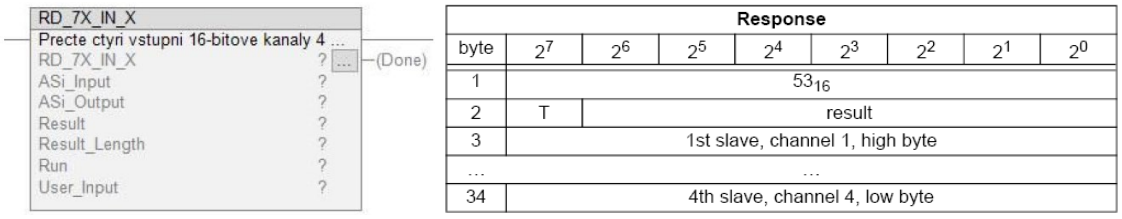
Obr. P4-3: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce RD_7X_IN_X

Pomocí této instrukce lze přečíst čtyři vstupní 16bitové kanály 4 ASi slejvů dle profilu 7.3 s adresami následujícími za sebou.

Tab. P4-4: Uživatelský vstup User_Input (SINT) RD_7X_IN_X

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		0	adresa 1. slejva				



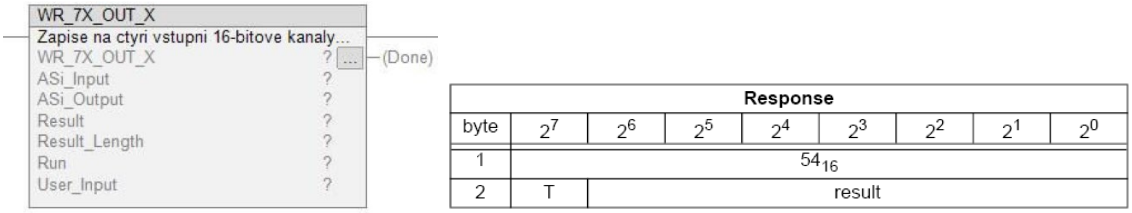
Obr. P4-4: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce WR_7X_OUT_X

Pomocí této instrukce lze zapsat na čtyři výstupní 16bitové kanály 4 ASi slejvů dle profilu 7.3 s adresami následujícími za sebou.

Tab. P4-5: Uživatelský vstup User_Input (SINT[33]) WR_7X_OUT_X

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		0	adresa 1. slejva				
2	1. slejv, kanál 1, horní byte							
…								
33	4. slejv, kanál 4, dolní byte							



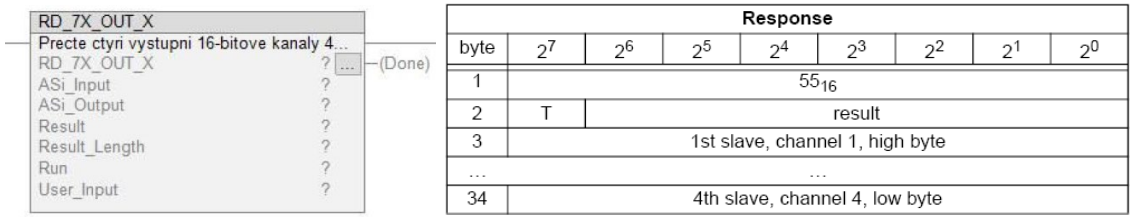
Obr. P4-5: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce RD_7X_OUT_X

Pomocí této instrukce lze číst čtyři výstupní 16bitové kanály 4 ASi slejvů dle profilu 7.3 s adresami následujícími za sebou.

Tab. P4-6: Uživatelský vstup User_Input (SINT) RD_7X_OUT_X

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		0	adresa 1. slejva				



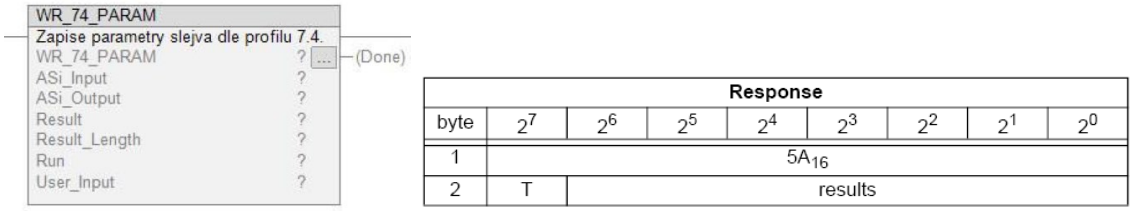
Obr. P4-6: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce WR_74_PARAM

S touto instrukcí lze zapsat parametrový řetězec slejva dle profilu S-7.4. Vzhledem k tomu, že řetězec může být delší, než je příkazové rozhraní, je částečně zapsán do bufferu, a poté zapsán na slejva. ‚n' je délka části řetězce, která se má zapsat do bufferu od indexu ‚i' dále.

Tab. P4-7: Uživatelský vstup User_Input (SINT[33]) WR_74_PARAM

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	adresa slejva							
2	i							
3	n							
4	buffer byte i							
	…							
n+5	buffer byte i+n-1							



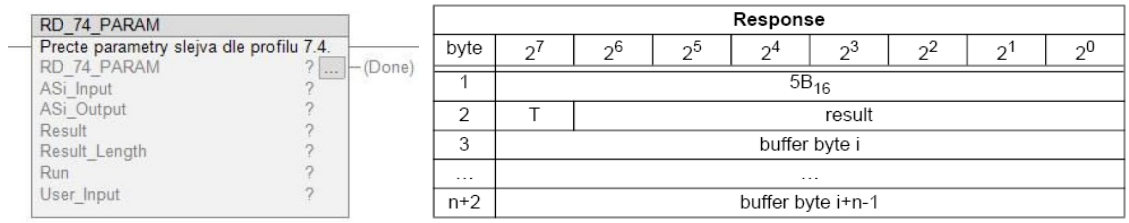
Obr. P4-7: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce RD_74_PARAM

S touto instrukcí lze číst parametrový řetězec slejva dle profilu S-7.4. Vzhledem k tomu, že řetězec může být delší, než je příkazové rozhraní, je částečně zapsán do bufferu. Obsah bufferu lze vyčítat v částech od indexu ‚i'.

Tab. P4-8: Uživatelský vstup User_Input (SINT[2]) RD_74_PARAM

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	adresa slejva							
2	i							



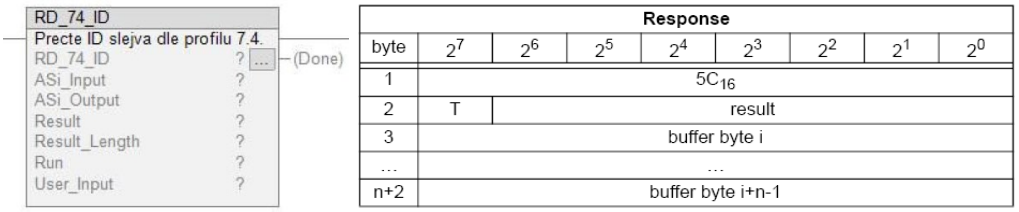
Obr. P4-8: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce RD_74_ID

S touto instrukcí lze číst ID řetězec slejva dle profilu S-7.4. Vzhledem k tomu, že řetězec může být delší, než je příkazové rozhraní, je částečně zapsán do bufferu. Obsah bufferu lze vyčítat v částech od indexu ‚i'.

Tab. P4-9: Uživatelský vstup User_Input (SINT[2]) RD_74_ID

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	adresa slejva							
2	i							



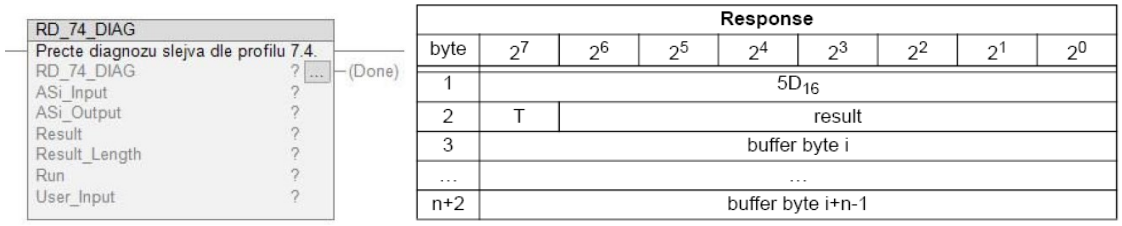
Obr. P4-9: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce RD_74_DIAG

S touto instrukcí lze číst diagnostický řetězec slejva dle profilu S-7.4. Vzhledem k tomu, že řetězec může být delší, než je příkazové rozhraní, je částečně zapsán do bufferu. Obsah bufferu lze vyčítat v částech od indexu ‚i'.

Tab. P4-10: Uživatelský vstup User_Input (SINT[2]) RD_74_DIAG

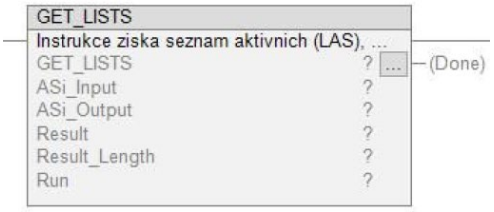
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	adresa slejva							
2	i							



Obr. P4-10: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_LISTS

Instrukce získá seznam aktivních (LAS), detekovaných (LDS) a projektovaných (LPS) slejvů a seznam příznaků (FLAGS) na základě specifikace ASi slejva.

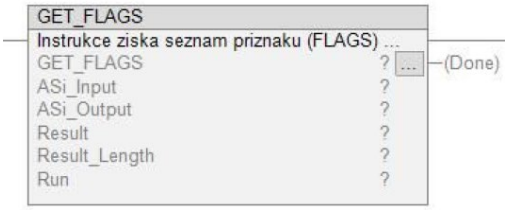


Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	30_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
…	LAS							
10	31B	30B	29B	28B	27B	26B	25B	24B
11	7A	6As	5A	4A	3A	2A	1A	0A
…	LDS							
19	31B	30B	29B	28B	27B	26B	25B	24B
20	7A	6As	5A	4A	3A	2A	1A	0A
…	LPS							
26	31B	30B	29B	28B	27B	26B	25B	24B
27	–							Pok
28	OR	APF	NA	CA	AAv	AAs	S0	Cok
29	–					AAe	OL	DX

Obr. P4-11: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_FLAGS

Instrukce získá seznam příznaků (FLAGS) na základě specifikace ASi slejva.

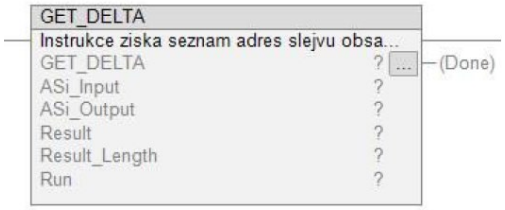


Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	47_{16}							
2	T	response						
3	–							Pok
4	OR	APF	NA	CA	AAv	AAs	S0	Cok
5	–					AAe	OL	DX

Obr. P4-12: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_DELTA

Instrukce získá seznam adres slejvů obsahující konfigurační chyby. Seznam vytváří na základě porovnání seznamu aktivních (LAS), detekovaných (LDS) a projektovaných (LPS) slejvů.



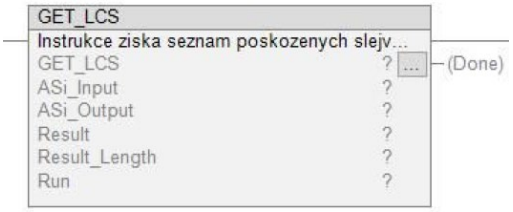
Response (if O ≡ 0)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	57_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	–
…	…							
10	31B	30B	29B	28B	27B	26B	25B	24B

Response (if O ≡ 1)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	57_{16}							
2	T	result						
3	0	1A	2A	3A	4A	5A	6A	7A
…	…							
10	24B	25B	26B	27B	28B	29B	30B	31B

Obr. P4-13: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_LCS

Instrukce získá seznam poškozených slejvů (LCS) na základě historie seznamu delta.

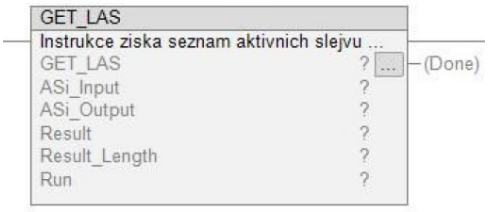


Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	60_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
…	…							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. P4-14: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_LAS

Instrukce získá seznam aktivních slejvů (LAS).

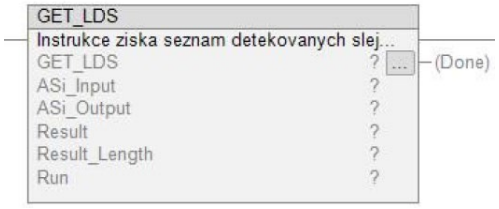


Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	45_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
…	…							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. P4-15: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_LDS

Instrukce získá seznam detekovaných slejvů (LDS).

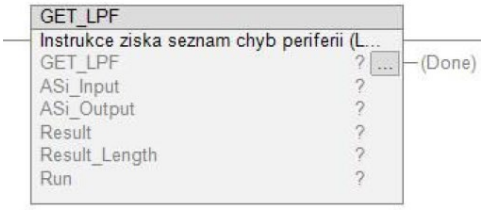


Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	46_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
…	…							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. P4-16: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_LPF

Instrukce získá seznam chyb periferií (LPF) signalizovaných ASi slejvy. LPF se obnovuje cyklicky ASi masterem. Kdy slejv signalizuje chybu, lze zjistit z popisu ASi slejva.

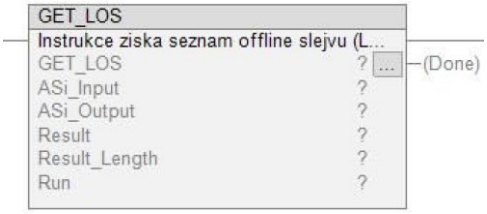


Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$3E_{16}$							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
...	...							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. P4-17: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_LOS

Instrukce získá seznam offline slejvů (LOS). Takový slejv způsobuje offline fázi, když se objeví konfigurační chyba.



Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	61_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
...	...							
10	31B	30B	29B	28B	27B	26B	25B	24B

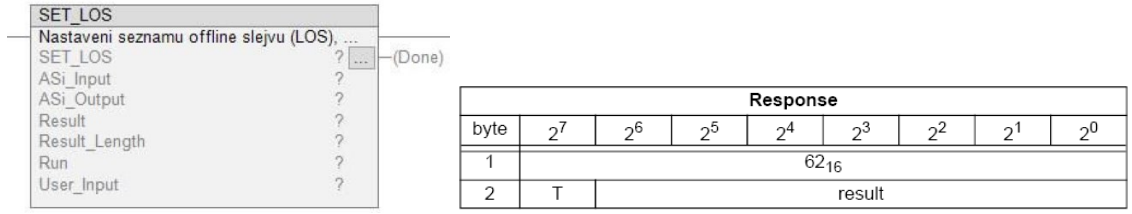
Obr. P4-18: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce SET_LOS

Instrukcí lze definovat seznam offline slejvů (LOS), tj. slejvy, které způsobují offline fázi, když se objeví konfigurační chyba. Uživatel si může zvolit, jak bude master reagovat. Master může být přepnut do offlinu, když nastane konfigurační chyba důležitého slejva, u méně důležitých slejvů lze informovat o chybě bez přepnutí do offlinu.

Tab. P4-11: Uživatelský vstup User_Input (SINT[8]) SET_LOS

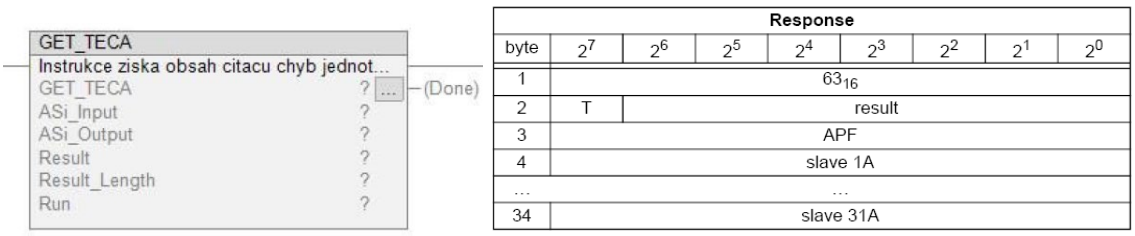
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	7A	6A	5A	4A	3A	2A	1A	0A
				...				
8	31B	30B	29B	28B	27B	26B	25B	24B



Obr. P4-19: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_TECA

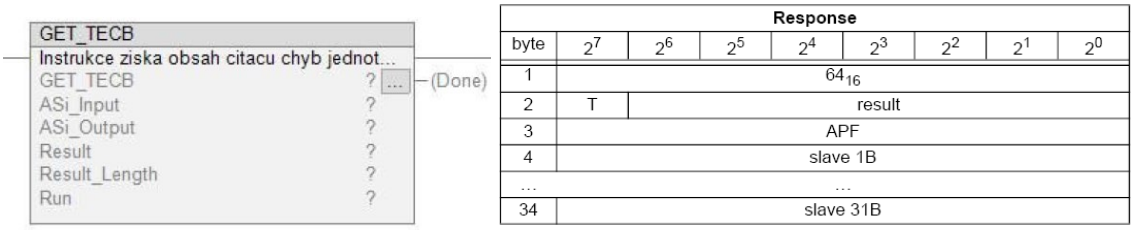
Instrukce získá obsah čítačů chyb jednotlivých slejvů/A-slejvů. Každým přečtením čítačů jsou čítače resetovány. Hodnota čítače je omezena na 254. Číslo 255 způsobí přetečení čítače.



Obr. P4-20: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_TECB

Instrukce získá obsah čítačů chyb jednotlivých B-slejvů. Každým přečtením čítačů jsou čítače resetovány. Hodnota čítače je omezena na 254. Číslo 255 způsobí přetečení čítače.

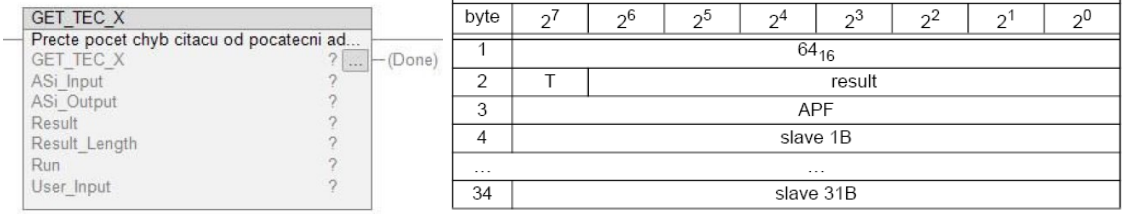


Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	64_{16}							
2	T	result						
3	APF							
4	slave 1B							
...	...							
34	slave 31B							

Obr. P4-21: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_TEC_X

Instrukce získá obsah čítačů chyb začínající od definované adresy. Každým přečtením čítačů jsou čítače resetovány. Hodnota čítače je omezena na 254. Číslo 255 způsobí přetečení čítače.



Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	64_{16}							
2	T	result						
3	APF							
4	slave 1B							
...	...							
34	slave 31B							

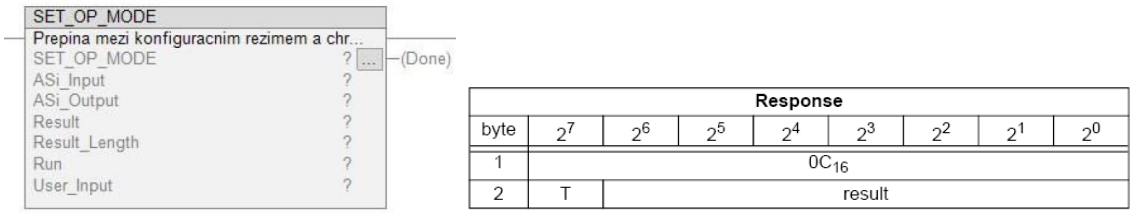
Obr. P4-22: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce SET_OP_MODE

Instrukce přepíná mezi konfiguračním módem a ochranným (protekčním) módem. V ochranném módu (OP.M. = 0) jsou aktivováni pouze slejvové, kteří jsou v seznamu projektovaných a shodují se jim konfigurace. Konfigurační mód (OP.M. = 1) aktivuje všechny detekované slejvy (kromě nulového slejva „0“).

Tab. P4-12: Uživatelský vstup User_Input (SINT) SET_OP_MODE

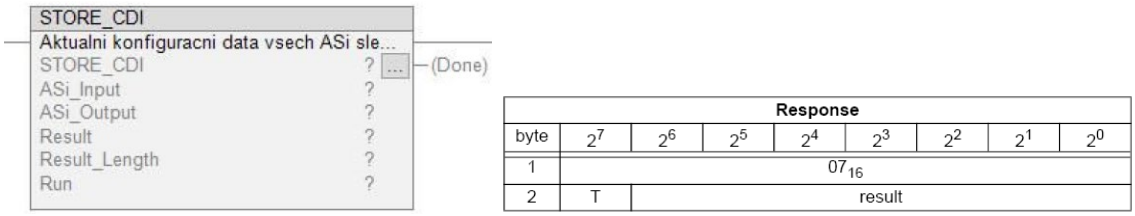
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	operační mód							



Obr. P4-23: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce STORE_CDI

Instrukce uloží permanentně na EEPROM aktuální konfigurační data všech ASi slejvů. Seznam aktivovaných slejvů (LAS) je přebrán do seznamu permanentních slejvů (LPS).



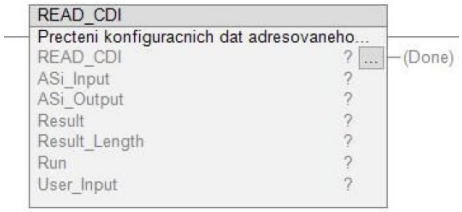
Obr. P4-24: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce READ_CDI

Instrukce přečte aktuální konfigurační data adresovaného ASi slejva. Dojde k přečtení I/O konfigurace, ID kódu a rozšířeného ID1 kódu a ID2 kódu. Pokud je B = 0, bude brán jeden ASi slejv nebo A-slejv. B = 1 znamená B-slejv.

Tab. P4-13: Uživatelský vstup User_Input (SINT) READ_CDI

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		B	adresa slejva				



Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	28_{16}							
2	T	result						
3	xID2				xID1			
4	ID				I0			

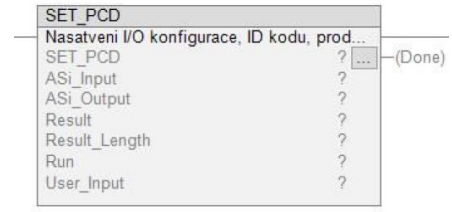
Obr. P4-25: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce SET_PCD

Instrukce nastaví permanentní konfiguraci adresovanému ASi slejvovi. Jedná se o I/O konfiguraci, ID kód a rozšířený ID1 kód a ID2 kód. Konfigurace bude uložena v permanentní paměti EEPROM. B = 1 znamená B-slejv.

Tab. P4-14: Uživatelský vstup User_Input (SINT[3]) SET_PCD

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		B	adresa slejva				
2	xID2				xID1			
3	ID				IO			



Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	25_{16}							
2	T	result						

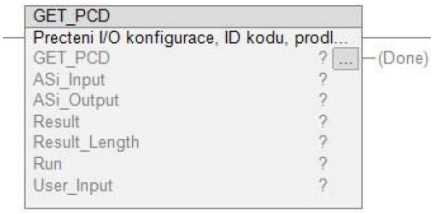
Obr. P4-26: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_PCD

Instrukce získá rozšířenou permanentní konfiguraci adresovaného ASi slejva. Dojde k přečtení I/O konfigurace, ID kódu a rozšířeného ID1 kódu a ID2 kódu. Pokud je B = 0, bude brán jeden ASi slejv nebo A-slejv. B = 1 znamená B-slejv.

Tab. P4-15: Uživatelský vstup User_Input (SINT) GET_PCD

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		B	adresa slejva				



Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	26_{16}							
2	T	result						
3	xID2				xID1			
4	ID				I0			

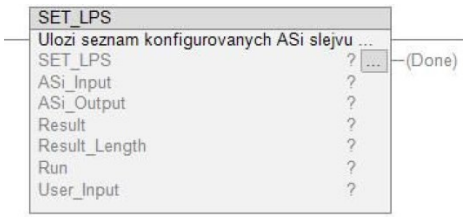
Obr. P4-27: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce SET_LPS

Instrukce převede seznam konfigurovaných ASi slejvů do permanentního úložiště EEPROM ASi mastera.

Tab. P4-16: Uživatelský vstup User_Input (SINT[9]) SET_LPS

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	00_{16}							
2	7A	6A	5A	4A	3A	2A	1A	0A
...								
9	31B	30B	29B	28B	27B	26B	25B	24B

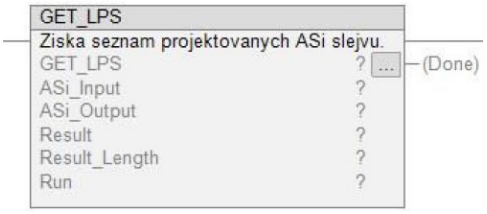


Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	29_{16}							
2	T	result						

Obr. P4-28: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_LPS

Instrukce získá seznam projektovaných ASi slejvů v permanentním úložišti EEPROM ASi mastera.

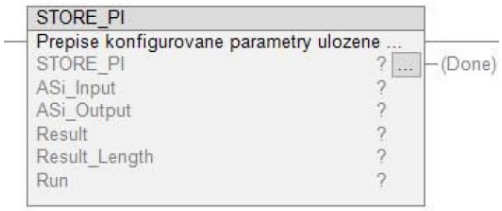


Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	44_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
...	...							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. P4-29: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce STORE_PI

Instrukce přepíše uloženou konfiguraci v EEPROM aktuální konfigurací. Jednoduše řečeno uloží aktuální parametry.



Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	04_{16}							
2	T	result						

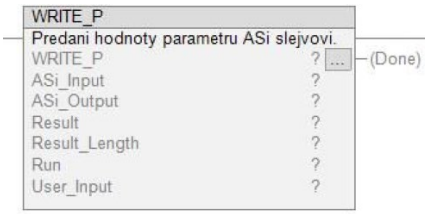
Obr. P4-30: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce WRITE_P

Instrukce zapíše parametr ASi slejva na adresovaného ASi slejva. Tento parametr je pouze dočasný a neuloží se do permanentního úložiště EEPROM.

Tab. P4-17: Uživatelský vstup User_Input (SINT[2]) WRITE_P

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		B	adresa slejva				
2	-				parametr			



Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	02_{16}							
2	T	result						
3	–				slave response			

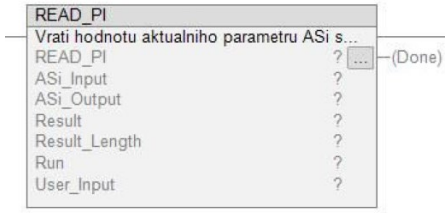
Obr. P4-31: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce READ_PI

Instrukce získá hodnotu aktuálního parametru ASi slejva. Tuto hodnotu si nelze plést s parametrem echo (ozvěny), který zasílá ASi slejv jako odpověď na úkon write_p.

Tab. P4-18: Uživatelský vstup User_Input (SINT) READ_PI

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		B	adresa slejva				



Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	03_{16}							
2	T	result						
3	–				PI			

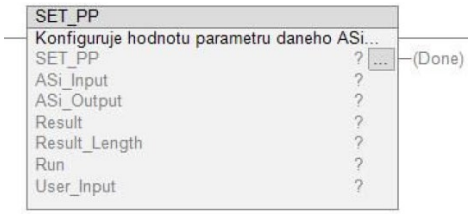
Obr. P4-32: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce SET_PP

Instrukce zapíše parametr do konfigurace adresovaného ASi slejva. Hodnota je permanentně uložena v EEPROM ASi mastera.

Tab. P4-19: Uživatelský vstup User_Input (SINT[2]) SET_PP

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		B	adresa slejva				
2	-				PP			



Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	43_{16}							
2	T	result						

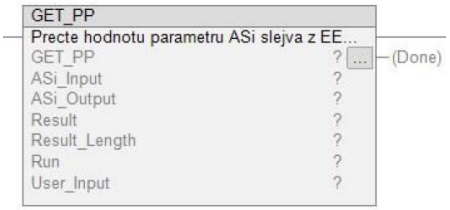
Obr. P4-33: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce GET_PP

Instrukce získá hodnotu parametru ASi slejva uloženou v EEPROM ASi mastera.

Tab. P4-20: Uživatelský vstup User_Input (SINT) GET_PP

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-		B	adresa slejva				



Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	01_{16}							
2	T	result						
3	–				PP			

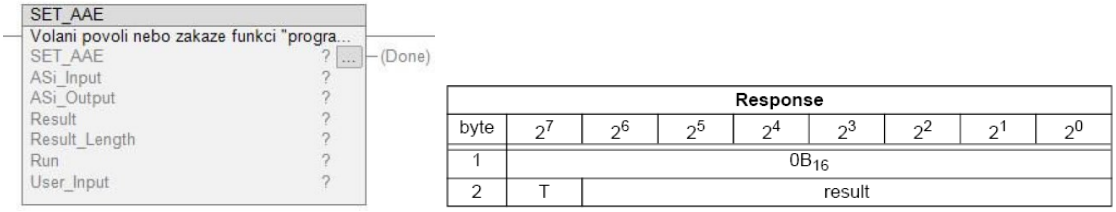
Obr. P4-34: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce SET_AAE

Instrukce povolí (AAE = 1), nebo zakáže funkci automatické adresace (AAE = 0). Tento bit je uložen permanentně, tedy nelze jej vrátit restartem ASi mastera.

Tab. P4-21: Uživatelský vstup User_Input (SINT) SET_AAE

Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	AAE							



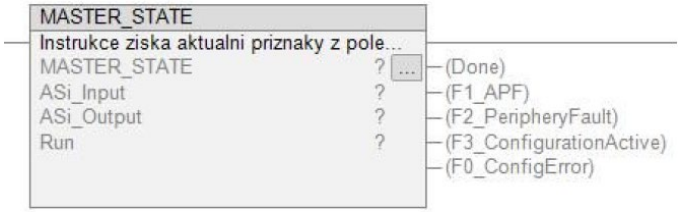
Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$0B_{16}$							
2	T	result						

Obr. P4-35: Add-On instrukce (vlevo) a pole odpovědi (vpravo) [23]

Instrukce MASTER_STATE

Instrukce neobsahuje žádné pole požadavku ani pole odpovědi, nejedná se o tradiční příkaz. Jejím voláním lze na čtyřech výstupech typu BOOL přečíst příznaky ASi mastera. V případě BWU1488 si instrukce interně zjistí, zda se jedná o síť 0, nebo 1, a podle toho čte správný úsek v poli.

- F0 – ConfigError: 0 = ConfigOK, 1 = ConfigError
- F1 – APF: 0 = ASi Power OK, 1 = ASi Power Fail
- F2 – PeripheryFault: 0 = PeripheryOK, 1 = PeripheryFault
- F3 – ConfigureActive: 0 = ConfigureActive, 1 = ConfigureInactive



Obr. P4-36: Add-On instrukce